DESIGN AND LAYOUT OF A DATA PATH FOR A 16-BIT PROCESSOR

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

By

M. S. BABU

to the
DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
JULY, 1983

ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude to my thesis supervisor, Dr. R. Raghuram, for his sustained interest in the work, constant encouragement, and guidance.

Kanpur
July, 1983

M.S. BABU

CERTIFICATE

Certified that this work titled 'DESIGN AND LAYOUT OF A DATA PATH FOR A 16-BIT PROCESSOR' by Mr. Mandava Surendra Babu has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

R. Raghuram

Dr. R. Raghuram
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology
Kanpur, India.

June, 1983

EE-1983-M-BAB-DES

ABSTRACT

Design of VLSI/LSI circuits is rather complicated. structured design approach, proposed by Mead and Conway analysed. This design methodology simplifies the task human designer and normally gives rise to designs that perform well as far as time delay, power dissipation ar silicon area are concerned. The computer aided design tolls are briefly described.

The design approach is illustrated by designing a path unit for a 16-bit processor. A detailed circuit is given. The major subsystems of the data path unit arithmetic logic unit, register array, barrel shifter the bus interface unit. Carry look ahead is implement the arithmetic logic unit by a simple carry bypass. register array has 16 dual port registers. Two intern busses run through the data path. Communication betwe the two internal busses and the system bus is controll the bus interface unit. The technology is assumed to

The layout of some of the basic cells is shown in of the stick diagrams. The 'Gate matrix layout' metho used to describe the layouts.

TABLE OF CONTENTS

CHAPTER 1

ANALYSIS OF THE STRUCTURED DESIGN METHODOLOGY

The complexity of LSI/VLSI design requires a new design methodology which is different from the one used in designing SSI circuits. A structured design methodology [1] is studied.

1.1 WHAT IS IT ABOUT?

It is one which supports hierarchy and regularity thereby giving rise to designs which usually have minimum power dissipation and delay.

It also allows the designer to take advantage of the architectural possibilities offered by the technology. It tries to constrain the complexity by producing designs with simple and regular interconnection topology. This approach simplifies the task of layout and modification.

1.2 WHY SHOULD ONE OPT FOR A NEW DESIGN APPROACH?

VLSI technology can produce chips containing a hundred thousand transistors. To design circuits of that complexity one must search for new design strategies. The traditional logic design methodology is unstructured and results in chip designs of great geometrical and topological

complexity, relative to its processing power.

The traditional switching theory helps us to implement a function with minimum number of gates. But at the VLSI level the area occupied on the silicon surface by a circuit is more a function of topological properties of circuit interconnection than it is of the number of logic gates implemented. Although, switching theory directly synthesizes the logic circuit design and gives a minimum gate implementation it does not give any information on the lower bound on area, power and delay time for the logic circuit.

So, there is need for a new design approach which not only constraints the complexity but also minimizes the delay, area and power dissipated for implementing a logic function.

1.3 DESIGN PHILOSOPHY

The design methodology has two basic parts. One is hierarchy and the other is regularity. Hierarchical techniques have long been used to design complex systems. Hierarchies are used to partition designs among the design team members. They also make sure that the common parts of the design are factored out and specified only once.

The design can be reduced in complexity by introducing regularity into it. This stems from the fact that the subunits are replicated many times and connection between units is simplified. Examples of systems which have a regular

structure are the ROM and PLA (Programmable logic array). Wiring strategy and regularity must be addressed from the start, eliminating inefficient and costly routing. This can be best done by providing proper feedback among the architectural, circuit design and layout levels.

1.3.1 The Separated Hierarchy

The design proceeds in a top-down manner in which the problem is decomposed and refined. The designer is limited in the kinds of structures he may use to implement a certain function. The advantage is that the design can be implemented quickly and reliably.

The separated hierarchy (Fig. 1.1) has two kinds of cells; leaf cells and composition cells [2]. A leaf cell is the most basic cell which is defined only in terms of primitives. No instances of other cells are allowed. A composition cell contains only logical interconnections of instances of other cells, no primitives are allowed.

Composition cells in the hierarchy form a representation independent language for specifying a design. A representation is one particular view of a design. The typical levels of representation are layout, stick-diagram, circuit diagram, behavioural description etc. Note that the leaf cells must be specified for each representation because they contain primitives. The same composition cells can be used for all

representations since they have no primitives. One must make sure that while representing leaf cells in different ways the consistency is not lost.

1.3.2 Algorithmic design and parameterized cells

Large chips resemble large programs in variety and complexity. The designs can be represented as programs. This allows placement of features to be done in a relative way, so that if one item or cell moves or grows, others follow.

It is very useful to parameterize cells so that they can be adopted to the environment in which they are instantiated . This kind of cells are useful building blocks in designs which may have different requirements. They also allow decisions about detailed characteristics of the cell to be delayed until later in the design cycle.

Consider a cell which is used in different environments in a particular design. Assume that the cell has 'n' available functions. But each usage of the cell may require a different subset of the 'n' available functions. Since the cells are represented as programs, we need not design all 2^n possible implementations of the cell and select the appropriate one. Instead, we can generate one cell program that can remove unnecessary circuitry and generate a cell with required characteristics. So, parameterized cells,

defined algorithmically can adopt to changes by restricting the small changes in design to small amount of effort by designers to incorporate these changes.

1.4 DOMAINS OF DESIGN DESCRIPTION

It is possible to identify three domains of design description which must be addressed in a finished design [2]. They are :

1) Behavioural description
2) Structural description
3) Physical description

1.4.1 Behavioural description

It is the description of the design. An integrated circuit must have a well defined behaviour. A design which does not have the desired behaviour is useless,no matter how clever the design. In this respect designing hardware and software are similar. The solutions to the problems in these two kinds of designs are also similar. Designs are structured, hierarchical and divided functionally into meaningful pieces. Tools are needed to help convert the high-level description of the behaviour into a low-level implementation description.

1.4.2 Structural description

A design is not merely a behaviour. Designing integrated circuits is a mapping of behaviour into physical structure. But there are some fundamental differences between software

design and integrated circuit design. One cannot implement a function as a chip without addressing the physical implementation issues. The designs, as well as the design methodology must address the physical aspects of the medium.

Integrated circuit design has a limited communication space which shares the computation space on the silicon surface. Thus, communication costs are high in silicon and this must be taken into account in designing integrated circuits.

The structural description is a description of the logical connection of blocks in the system. Hierarchical decomposition of behaviour of the design into blocks is done along both geometrical and functional lines. The logical connection, interface between functional units is precisely along the geometrical interfaces.

The hierarchical decomposition is driven by a very high level part-behavioural, part-physical floor plan. The floor plan is a general functional decomposition strategy which includes a wiring strategy as part of decomposition. A good floor plan recognizes the two-dimensional nature of the silicon chip and addresses physical problems such as global wiring.

1.4.3 Physical description

The physical characteristics of VLSI chips introduce difficulties in complexity management like the production of geometrical structure under many constraints of topology and physics. For large designs the physical constraints of the silicon medium and the communication limitations of silicon must be addressed early in the design. Systems which deny the physical nature of silicon implementation cannot effectively use the silicon in large designs. Stick diagrams can be used to generate geometrical layouts, without the need for design rule checking. This way designs can be produced much more quickly and more iterations on a design can be quickly done to produce an optimized layout.

1.5 DESIGN PROCESS

The design process embodies the structured design methodology described above. The design process has two distinct parts : design and implementation.

Design proceeds top-down with global decisions made first. The implementation then proceeds bottom-up where constraints from low-level implementations are propogated to higher levels in the design hierarchy.

1.5.1 Overview of design process

The design process is divided into five parts : architectural design, where the design is partitioned into functional blocks and the general floor plan of the design is decided. Cell estimation, where cell interfaces size and interconnections are decided; cell detailing, where detailed cells are laid out. Chip integration, where cells are assembled into chips. Fabrication, where the finished design is converted into a form suitable for fabrication equipment. Note that the design process may iterate in any of the loops seen in Fig. 1.2.

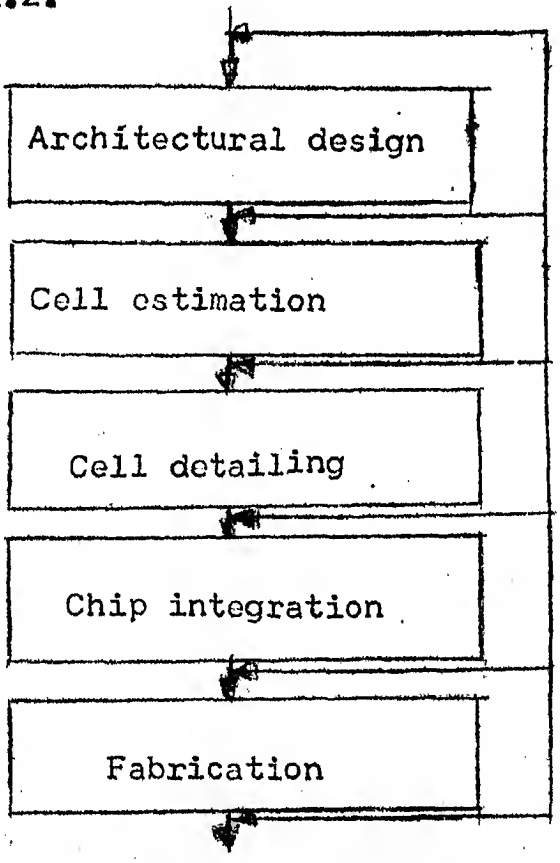


Fig. 1.2 Design process flow chart

Now, each step in the design process is studied in detail.

1.5.2 Architectural design

The flow chart is shown in Fig. 1.3.

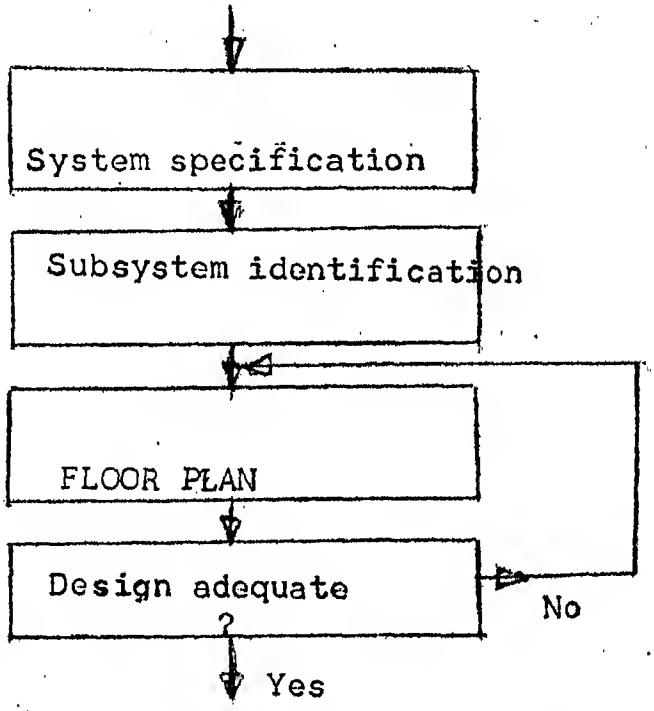


Fig. 1.3 Finish architectural design

The high-level top-down architectural design is still predominantly a human task. Efforts are being made to automate this part of the design too. For regular structures like ROM, PLA automation can be quite useful.

1.5.2.1 System specifications

It is the functional description of the system. The description must unambiguously specify what the system is expected to do.

1.5.2.2 Subsystem identification

The functional decomposition is done here. Each subsystem will perform a function. The function of each subsystem must be specified clearly.

1.5.2.3 Floor plan

The designer must not only concern himself with functional decomposition but with wiring strategy as well. The product of architectural design is a floor plan, a tiling of the plane with functional units. The floor plan includes a rough specification for each of its elements, including constraints interfaces and desired features. It also takes into account the communication problems of two-dimensional silicon. At this point, expected critical paths, both in area and speed can be estimated. If the design is not found adequate at this stage then we go back to the floor plan stage and modify the design. The clocking scheme must also be decided at the architectural level.

1.5.3 CELL ESTIMATION

Here, the basic cells are identified (both composition and leaf cells). Approximate geometries are tried for the

blocks defined in the floor plan. If these designs are difficult to make or optimize then we go back to the floor plan stage and modify the design. Stick diagrams are a useful notation in this phase since they allow both structural and geometrical information to be expressed in a highly readable form. The cell estimation phase proceeds until all interfaces between cells are completely reconciled.

We may plan our digital processing systems of combinations of Register-to-Register data transfer paths controlled by finite-state machines. Then the geometric shapes, relative sizes and interconnection topologies of all subsystem modules are planned so that all modules will merge together snugly, with a minimum of space and time wasted by random wiring. Storage registers are constructed by using charge stored on input gates of inverting logic. The combinational logic in the data-paths is implemented using steering logic composed of regular structure of pass-transistors. The combinational logic in the finite-state machines may be implemented using PLA's. All functioning is sequenced using a two phase non-overlapping clock scheme. The entire system may be viewed as a giant hierarchy of nested machines, each level containing and controlling the level below it. Evaluation of routing parasitics is also done.

1.5.4 Cell detailing

In this phase, final designs are produced for each low-level cell using graphic or program oriented design aids. Detailed cells may be generated in the form of hard mask geometry, malleable sticks or algorithmically defined cells, cell detailing is the start of bottom-up implementation of design. In large systems, most cells are not in the critical path for speed or area used and so do not require optimization. System which automate the layout are desirable considering the complexity of layout. A cell design must be consistent with floor plan and it should have the ability to interface to high-level assembly tools.

After the detailed layout of each cell one must verify the layout through a DRC (Design Rule Checker) program and one may also do the Timing Verification.

Stick diagrams are an important means for the generation of detailed cells as they guarantee designs free from geometrical design rule violations. They also provide a good interface to chip integration phase that follows.

1.5.5 Chip integration

It is the phase in which cells are assembled, according to floor plan, into a finished design. The assembly task can be made easy by having a good floor plan and well defined cell interfaces. Programs like chip assemblers and silicon

compilers exist, which, working with a given floor plan and properly defined cells, will assemble low-level cells into complete integrated systems.

Programming language based systems are preferred tools for the chip integration phase because of their versatility. Powerful compositions can be easily defined in this algorithmic manner. Properly defined compositions allow changes to individual cells to be made without requiring a change in the way those cells are composed into the system. New ideas and optimizations, as well as bug fixes, can be made without requiring large changes to the composition of the system.

1.5.6 Fabrication

This is the last step in the design process. This is typically a batch process, requiring a large processor for vast amounts of time. The hierarchical structures enables this phase of design to proceed fast. Plotters, mask generation programs and design rule checkers exist which take advantage of the hierarchy in the design to speed up processing considerably [3].

1.6 OUTLINE OF THE THESIS

In this thesis, we rigorously apply the design methodology described above, in the design of a data path for a 16-bit processor.

Chapter 2 gives a brief description of the CAD (Computer aided design) tools, which may be helpful in the design process.

Chapter 3 describes the design of the ALU (Arithmetic logic unit). Chapter 4 contains the design details of the remaining subsystems of the data path unit.

Finally, the conclusion is presented in Chapter 5.

CHAPTER 2

CAD TOOLS FOR LSI/VLSI

In this chapter we take a brief look at the various CAD (Computer aided design) tools that may be helpful at the various stages of IC design.

2.1 WHY CAD AND WHAT IS CAD?

The everincreasing complexity of integrated circuits demands the use of computer as an essential tool for designing large scale integrated circuits. These tools may range from interactive graphics and digitizing systems to individual programs used for circuit or logic simulations, mask layout, and data manipulation or reformatting. They form a set of software tools to provide the designer with design assistance during each phase of the design.

2.2 DESIGN SPECIFICATION

The structure and behaviour of a digital system must be described in various ways at many levels to completely characterize a design. But at present the existing languages are usually associated with specific descriptions of architecture, system behaviour, system structure, logical structure, circuit structure, logical behaviour, or physical structure. These descriptions lack the generality required to support

VLSI design [4]. Thus there is a need for a design description language which allows description of functional and physical entities and relationship among entities.

2.2.1 Functional description and synthesis

The overall behaviour of the system is specified at the architectural level. Then the behaviour can be synthesized to generate the necessary hardware which satisfies the constraints specified by the behaviour. Synthesize tools are a set of programs, which when executed with the given behaviour as data generate the hardware as result. They can be used to synthesize regular structures like memory arrays PLA's and decoders. But till today the architectural design is mainly a human task.

2.3 SIMULATION

This is a dynamic verification of the behaviour of a system within an environment specified by the designer.

2.3.1 Logic simulation

This is a gate level simulation which solves the equivalent boolean equations with delay elements inserted between gates to account for signal timing. One has to specify the inputs and the way the gates are interconnected for a logic circuit. The output waveform will be produced by the simulation program by making use of the specified data. The program can also be used for test sequence evaluation and

fault coverage calculations [5].

2.3.2 Circuit simulation

We have to specify the topology of the circuit alongwith the devices that are used to interconnect the various nodes and if any active devices are used we should specify the detailed models to be used. These programs perform a detailed simulation and can give the voltages and currents at any node in the circuit. One such typical simulation program is SPICE (Simulation program with integrated circuit emphasis) [6].

2.3.3 Timing simulation

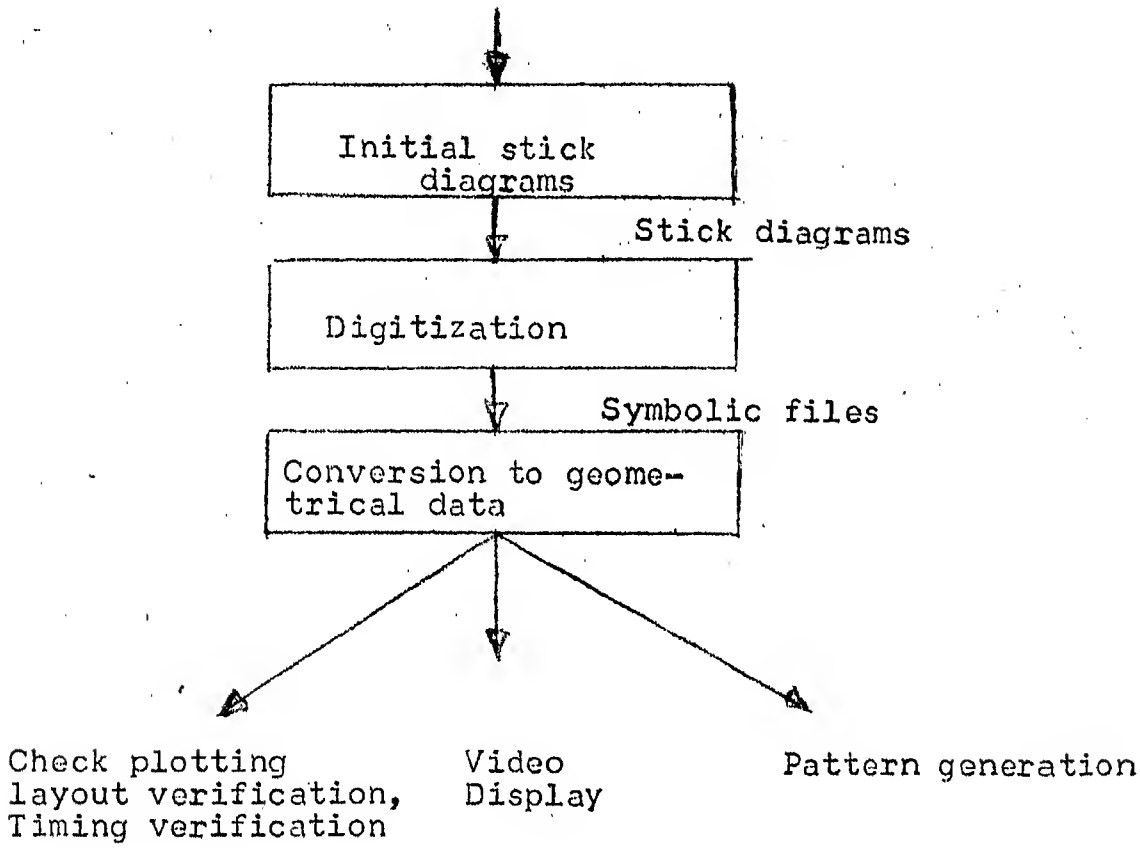
A timing simulator is in between the logic simulator and circuit simulator. It provides more accurate analysis than logic simulators and runs more efficiently than circuit simulators. A timing simulator uses only current-voltage tables for transistor models, capacitive loading, and circuit connectivity to determine signal waveforms at each circuit node. One typical timing simulator program is MOTIS (Timing simulator for MOS integrated circuits) [7].

2.4 LAYOUT

This is the most tedious and time consuming phase of IC design and hence it is well suited for computerization. The set of programs must be capable of constructing, editing, and reproducing complex figures, performing tolerance checks;

selective erase, expand, move and merge; taking symbolic input; pattern generation etc.

For each basic cell the layout phase may take the following sequence. (See Fig. 2.1).



2.4.1 Stick diagram representation of initial layout

It is useful to describe the initial layouts which are done manually on drawings with stick figures to represent transistors, polysilicon, metal and contact windows. One such method of coding could be dashed lines to represent polysilicon, thick lines to represent metal, boxes to represent transistors and dots to specify contact windows. If one has color plotting facilities then color coding can be done.

A stick diagram is a notation midway between transistor diagrams and full mask layouts. Stick diagrams specify more geometrical information than a transistor diagram in that the relative positions of transistors and wires are meaningful, but less than a full layout in that the absolute positions of transistors and wires are not meaningful [8]. This intermediate position has many advantages. The designer can specify his layout in a very sketchy manner, with no regard to exact positioning, yet still have control over the relative topology of the layout. Thus component recognition and circuit compaction free the designer from worries related to details about mask making. This abstraction allows the designer to concentrate on system design.

2.4.2 Conversion from stick diagrams to symbolic form

The manual layout is converted to computer readable symbolic representation by running an interactive digitization

program. The symbolic files created can be edited to correct layout errors or to implement logic updates.

2.4.3 Conversion from symbolic form to caltech intermediate form

The symbolic files are converted to CIF (caltech intermediate form) which is a hierarchical geometrical description, and contains commands to set the layer, construct rectangles, wires and polygons as well as facilities to define and call symbols.

2.4.4 Purpose of CIF

The CIF is a means of describing graphic items (mask features). Its purpose is to serve as a standard machine readable representation from which other forms can be constructed for output devices such as plotters, video displays, and pattern-generation machines. The form is a fairly readable text file, to simplify combining files. CIF thus serves as the common factor in the description of various integrated circuit projects. All the designs can be converted to CIF as an intermediate, before being translated again to a variety of formats for output devices or other design aids [1].

CIF has facilities to call and delete symbols. A symbol is a set of geometry and calls on other symbols, with an identification number and a scaling from CIF units to the symbols internal units.

2.5 LAYOUT VERIFICATION

2.5.1 Check plotting

At several points, the image of the designs, or of the entire die, must be viewed to detect any gross errors. Of course, if one had complete confidence in the accuracy of the tools that manipulate the artwork, check plotting may not be necessary at all. However, the cost of mask making and fabrication is high enough to make such checks worthwhile. A typical check plotting program may take CIF files as input and, in response to commands by the user, plots the various portions of the file's geometry [2], on a variety of output devices. It is essential that the designer be able to make small, localized modifications to a design and view the result quickly. Interactive views of individual cells and areas of the design must be available to the designer in a fairly short amount of time. This fast feedback is an important characteristic that allows small changes in the design to be viewed immediately regardless of the complexity of the layout.

2.5.2 Design rule checking

It is necessary that the design rules for layout which are specified by the process are not violated. The checking can make use of the symbol hierarchy found in the CIF description to eliminate as many redundant comparisons as possible [2]. The correctness of geometry inside a CIF

symbol is checked only once, regardless of how many instances of that symbol are made. The environment in which each instance of a symbol is found is remembered so that a particular set of symbol interaction is checked only once. This technique gains a great speed advantage on regular chip designs without restricting the complexity of the geometrical shapes taken as input.

The extraction of the design topology, a by product of DRC, can provide an important verification tool for designers in determining whether or not they have the circuits which they intended in the art work.

2.6 TRANSLATING CIF TO PATTERN GENERATOR FORMATS

A set of programs are used to convert CIF data to the formats required by pattern generator machines. These programs remove the CIF symbol hierarchy by recursively replacing instances of symbols with their geometrical primitives as it moves through the input file. Then it converts all the primitive CIF shapes into a set of rectangles available on the pattern generator.

The program also optimizes the ordering of data on the PG (pattern generator) tapes to minimize the time required by the PG machine in making the reticles. The PG output data produced can be visually checked by plotting it with a check-plotting program. This provides a picture of the final data as it is sent to the mask house.

CHAPTER 3

ARCHITECTURAL DESIGN OF DATA PATH AND THE DESIGN AND LAYOUT OF ALU

In this chapter, the architectural design of the data path unit is described. The design and layout of ALU (arithmetic logic unit) is also described. The structured design methodology is rigorously applied. The technology is chosen to be NMOS because of its richness of available circuit functions, topological properties of interconnection path etc.

3.1 WHAT IS A DATA PATH AND HOW DOES IT FIT INTO A COMPUTER SYSTEM?

The data path chip performs most of the data manipulation functions of the system. The operations are performed as directed by sequences of control micro-instructions which are fetched from the microcode memory and decoded by the microcode-decode control logic.

The data path chip along with the microcode-decode logic can be named as the execution unit of the system. One typical organization of a computer system could be as shown in Fig. 3.1.

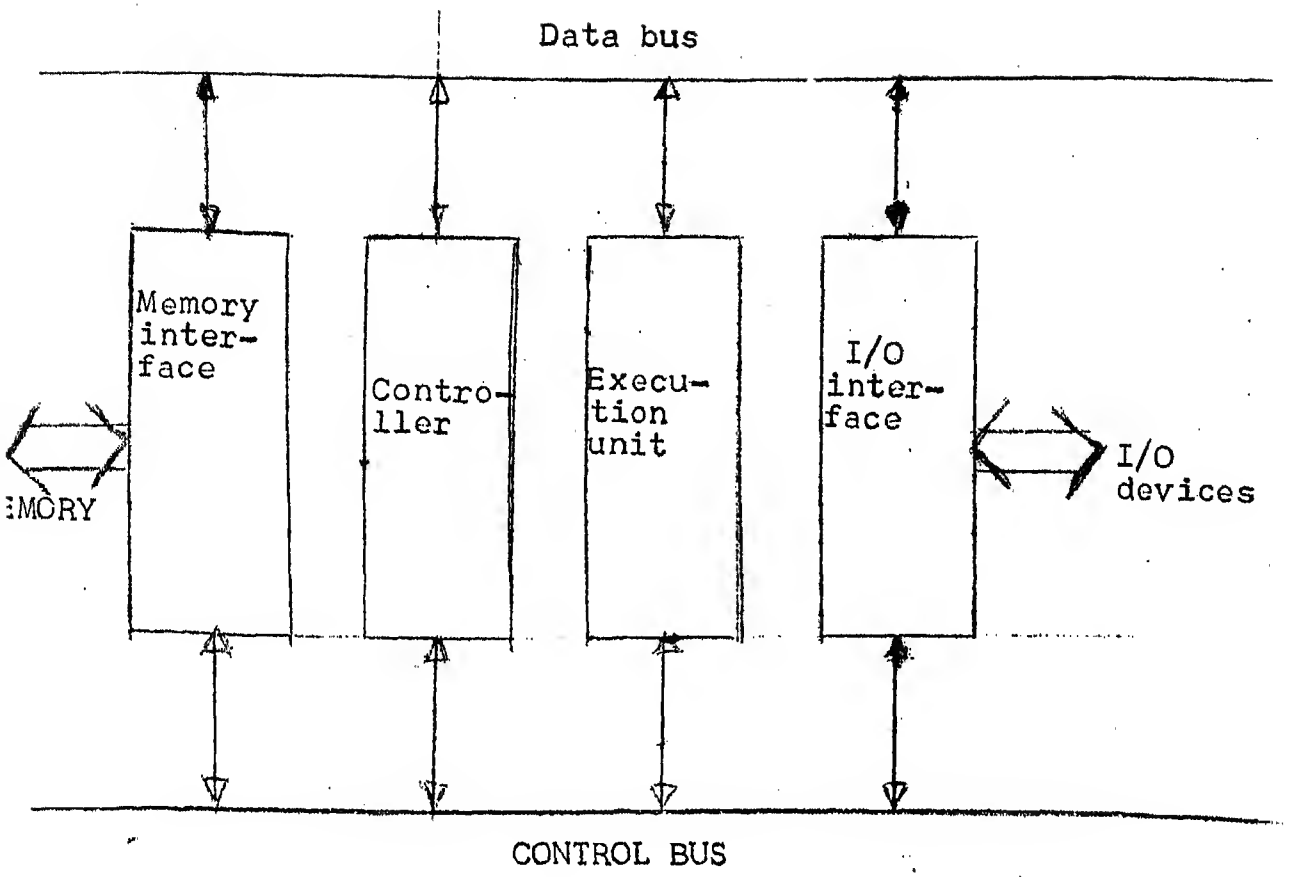


Fig. 3.1

The memory unit may have a certain amount of cache memory, capability to interface with the main memory unit, ability to implement the required data structures. It also stores the micro-program.

The controller chip can fetch the instructions from the cache, decode them and generate the necessary microcode memory address.

The execution unit can perform logical and arithmetic operations as per the directions given by the controller.

The I/O interface can communicate with the I/O devices.

All the subsystems are connected via a data bus and a control bus.

Now, we proceed to the first step in the design of the data path unit, the architectural design.

3.2 ARCHITECTURAL DESIGN

3.2.1 Identification of subsystems

Register file :

The data path unit should contain a set of registers, some general purpose and some special purpose to store information. We shall provide 16 registers, sufficient for a 16-bit CPU.

ALU :

The data path will obviously need an arithmetic logic unit to perform logical and arithmetic operations.

Barrel shifter :

In order to handle the variable length words, it also needs a shifter at least 16 bits long. There is a temporary register along with the shifter to store the shifter's output.

Bus interface unit :

The chip must include a bus control unit to communicate with the system bus for synchronous or asynchronous transfer of data.

ALU-shift register :

In order to support multiplication, the ALU result is latched in a 32-bit shift register.

Status register

Finally, there will be a 4-bit status flag register, which stores carry, sign, zero, overflow bit information.

3.2.2 Major Design Decisions

After having identified the subsystems, the organization of these on the chip is discussed.

Some of the important design goals are : 1) the performance of the system must be as fast as possible, 2) the unit must support microprogrammed control structure, thus allowing the instruction set to be chosen as per the application, 3) it must be able to do variable field operations for emulation instruction decoding, assembly of bit-maps for graphics etc.

The unit may have two internal busses so that two operands required for a particular operation can be acquired simultaneously. So, the register-file consists of

16 dual-port registers. A register can be read or written by either of the busses. The ALU may also have two-input latches to latch the two-operands simultaneously. The ALU output latch has access to both busses.

In order to speed up the ALU, a 4-bit carry look-ahead scheme is implemented by a simple carry bypass. There is also a control circuit for supporting multiplication. This enhances the speed of multiplication by eliminating the time consuming communication between execution unit and the controller.

The status information of the status-flag register is available at the external pins.

The barrel shifter must concatenate the two busses and then extract any consecutive 16 bits from the 32 bit word. Two phase non-overlapping clock scheme is employed.[1].

3.2.3 Derivation of floor plan

The chip floor plan is critical for evaluating alternative architectures, helping to determine the optimal arrangement of major functional modules, and resolving any basic interconnect problems, especially the routing of global signals such as power, ground and clocks. The structured design methodology recognizes wire and interconnect management as the basic problem of the design, one that must be addressed very early in the design process.

It may be useful to modify the architecture of a system to resolve any layout and performance problems. One can best experiment with this kind of trade-off at the floor plan level before doing any detailed layout. The penalty for not adjusting these trade-offs early is an integrated system that takes longer to layout, has reduced performance due to longer than necessary interconnect and occupies more area because the wiring and logic are not fully integrated.

The importance of treating MOS design as a wiring problem is illustrated by the fact that even the MOS transistors themselves result from the simple crossing of two interconnect layers diffusion and polysilicon. Thus, even the computational elements are reduced to a wiring problem at the implementation level.

Our floor plan is merely a block diagram with the blocks drawn to approximate scale and the routing of major busses, clocks, power, ground and critical signal paths specified in terms of their location and the layer on which they run.

The cells are interconnected by simple abuttment with their neighbours. The advantage is that it eases the design task by eliminating random wiring, uses space more efficiently by combining logic and busses and by eliminating

the extra space absorbed in intercell wire routing, and helps to improve performance by reducing interconnect lengths.

Keeping the above points in mind we first make the decisions that the two internal busses would run through the entire system from one end to another. The floor plan shown in Fig. 3.2 satisfies the constraints such as minimal interconnect wiring, optimum area and connection by cell abuttment.

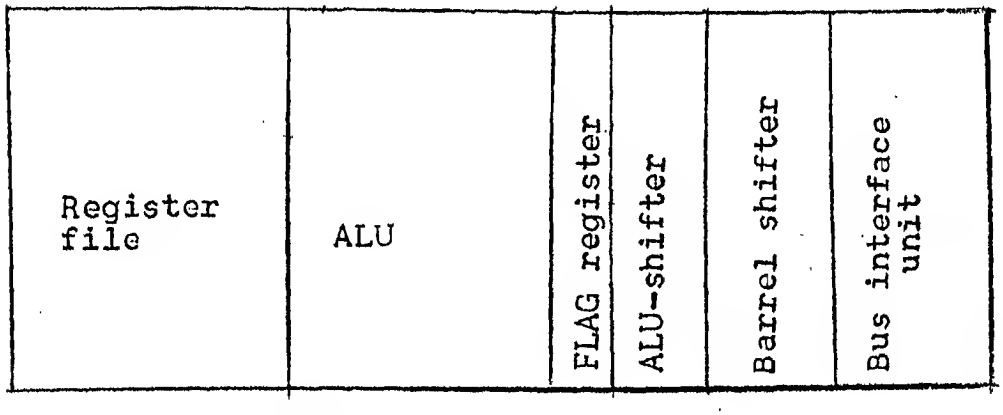


Fig. 3.2

Note that all the cells are of the same height which makes it easy to interconnect by abuttment.

The two internal busses are run on metal except in the ALU cell where they are run on polysilicon for reasons explained later. The power and ground lines are run on metal.

The internal busses while crossing the power and ground lines will run on polysilicon. The clock lines are usually run on polysilicon. In a particular cell, if the internal busses run on metal then the power lines will run parallel to the busses on metal.

Table 3.1 shows how the major busses, control lines, power lines are organized in each cell.

The VDD and GND net for the data path chip is shown in Fig. 3.3.

3.2.4 Clocking scheme

The clock has two non-overlapping phases φ_1 and φ_2 as shown in Fig. 3.4.

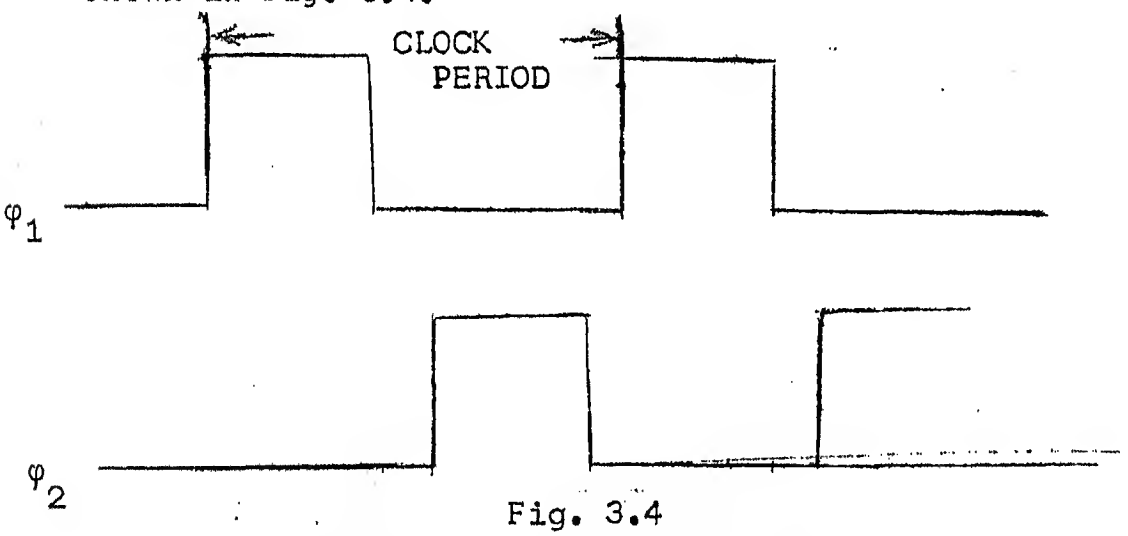


Fig. 3.4

During φ_1 (i.e., when φ_1 is high) data moves from one sub-system to another on the data path chip. Actually, the ALU input data is selected during φ_1. The typical data transfers that may take place during φ_1 include register to ALU input latch, external data to ALU latch, temporary register to ALU latch.

Table 3.1

Cell	Power	Ground	Clock and control lines	Internal busses
Register File	Metal horizontal	Metal horizontal	Poly vertical	Metal horizontal
ALU	Metal vertical	Metal vertical	Metal vertical	Poly horizontal
Bus Interface Unit, Shifter, ALU-Shifter, Status Register	Metal, horizontal	Metal horizontal	Poly vertical	Metal horizontal

The ALU operates during φ_2 (i.e., when φ_2 is high) and at the end of φ_2 the results are latched in the ALU output latch (ALU shifter).

The microcode that controls the ALU input selection may enter the data path chip during φ_2, i.e. the code will be latched in during φ_2 and will be active during φ_1. The microcode that controls the ALU operation may enter during φ_1 and will be active during φ_2.

The two phase clocking scheme is illustrated by taking a finite state machine as an example [1].

A single phase clocking scheme is discussed first.

A finite state machine is modelled as shown in Fig. 3.5.

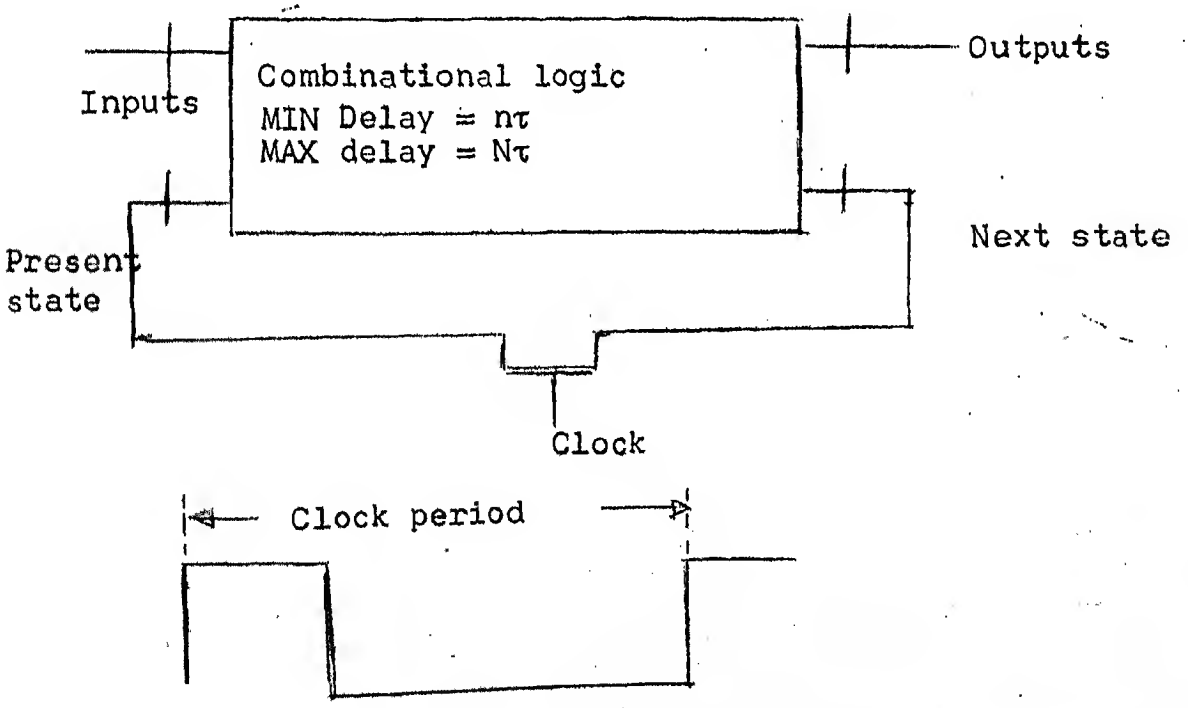


Fig. 3.5

This clocking scheme may be called the 'narrow pulse clocking scheme' because the clock width should be less than the minimum delay of the combinational logic ($n\tau$).

The present state information changes after a short time (Time taken to charge the gate capacitance) after the leading edge of the clock. Hence, the delay through the combinational logic must be greater than the clock width, or else the change in the present state information will propagate through the combinational logic to change the next state information before the trailing edge of the clock.

The combinational logic must also be designed to satisfy a two sided relation. It's delay must be greater than the clock width and less than the clock period.

These constraints make the 'narrow pulse clocking scheme' too risky and hence we go for a two phase clocking scheme. The clocking scheme shown in Fig. 3.6 includes four epochs. The first epoch is during φ_1 (when it is high) which must remain high long enough to charge the present state input nodes: this delay is named 'Delay time'. Following this delay the combinational logic starts setting up the outputs and next states, independent of when φ_1 may transit from high to low.

The second epoch is t_{12} because both the clocks (φ_1 and φ_2) must never be high simultaneously. This t_{12} may be

chosen to be as short as convenient. Note that during t_{12} the combinational logic is working and hence the system is not idle.

During the third epoch, when φ_2 is high, the clocked element samples its input. The outputs must be stable slightly before the trailing edge of φ_2, an interval called the preset time of the storage element. φ_2 must be wider than the preset time.

The last epoch is t_{21} (period of nonoverlap) during which the system is idle. So, it is necessary to keep t_{21} as small as possible to improve the system performance. But t_{21} serves the purpose of accommodating clock skew, a variation in the arrival time of the clock to different clocked storage elements.

The advantage of the two phase clocking scheme is that the clock period and its constituent epochs are, with static storage devices, involved in one sided relations in which a region of correct operation can always be found by making the epochs longer.

Before proceeding to the design of subsystems the layout strategy to be adapted is discussed in the next section.

3.3 LAYOUT STRATEGY

We will first discuss the attributes of a good layout strategy [9].

The conversion from logic design to mask artwork is arguably the most difficult part of the design of custom LSI/VLSI. There are number of considerations which must be taken into account in the development of a layout style which will suit the VLSI circuits. First of all the layout style must be able to make use of CAD tools in order to minimize the turn around time. The strategy should not waste much silicon area at the expense of layout ease because more the silicon area more the cost of implementation. The layout method must be adoptable to team effort. This is because the layout task is often divided among several designers, each working on different portions of the chip. So, the interconnections between different portions of the chip must also be easy.

The final requirement is that the layout should be easily updatable in order to accommodate new design rules. The checking of the layout should also be easy.

Now, the 'gate matrix layout' method which satisfies the above requirements is discussed. The layout strategy uses a regular, a matrix composed of intersecting rows and columns[9]. The columns are run on polysilicon level and

serve as transistor gates and interconnections. The rows are diffusion and at the intersection with a column form transistors.

The planning stage of the layout consists of making representational line drawing or stick diagram using the levels of interconnection available. For the NMOS poly-silicon gate technology, these levels are : polysilicon, metal, and diffusion. One can draw a series of parallel and equally spaced polysilicon lines which form the inputs (control the gates of the transistors) of the circuit. The output of one stage of a circuit may also be run on poly if it forms the input of a second stage. Thus the number of polysilicon lines may be greater than the number of discrete inputs to a circuit.

In the stick figure transistors will be drawn of rectangles on the gating polysilicon column. Subsequent transistor placements will be determined by two factors : 1) input column, 2) the association among the transistors. After the rows are defined, further interconnections are done with either metal or diffusion. These diffusions are drawn as single lines. Contact to either the diffusion or the polysilicon is represented by a dot. The metal can run as an interconnect in either directions (horizontal or vertical). The metal is represented by a thick line. The metal which runs in the horizontal direction has the same pitch as the rows.

Notation for the stick diagram is shown in Fig. 3.7.

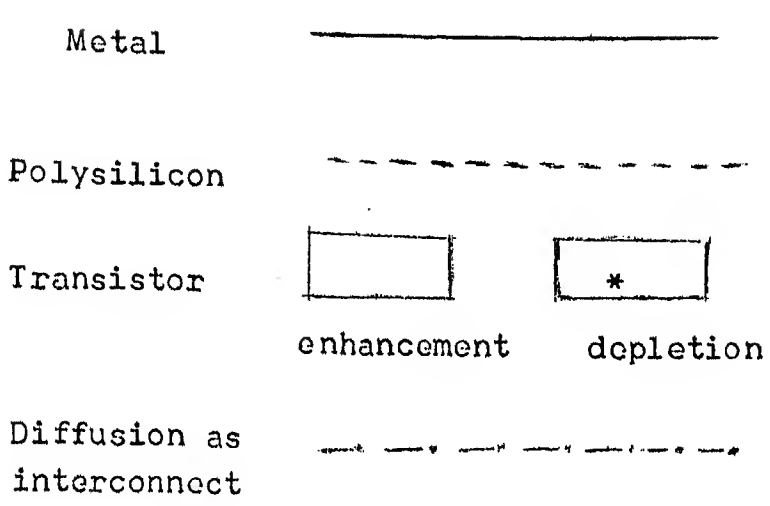


Fig. 3.7

Note that the pitch of the rows is determined by the minimum allowed distance between the two discrete transistors without cross coupling. The column pitch is bounded by the space required to accommodate a diffusion region with a contact window in between two polysilicon columns.

We have seen how a stick diagram can be drawn once a logic is designed. Now the task is to translate the stick diagram to final artwork. We have to digitize the stick diagram in order to make use of a computer to generate the

artwork. So we need a symbolic representation which takes full advantage of the regularity in the structure. Symbolic layouts are constructed by the placement of symbols on a grid which serve to create the topology for a given circuit. The number of symbols used varies according to the technology used. Each symbol represents geometries which may include any number of mask levels. The designer is relieved of the task of having to hand draw the actual mask geometries.

The nature of gate matrix, composed of intersecting columns of polysilicon and rows of diffusion, allows a high degree of simplicity both within the composites of the symbols and also in terms of the number of symbols required to describe the layout. More than one geometry can be represented by a symbol, unless there is an ambiguity.

Some of the characteristics of the gate matrix layout method are :

1) Polysilicon runs only in one direction.

2) Diffusion runners exist between polysilicon columns.

3) Metal runs in both directions and is constant in width except for power busses.

4) Transistors can exist only on polysilicon columns.

The following is a description of the set of symbols that are used in gate matrix layout method for NMOS technology.

E transistor in enhancement mode

D transistor is depletion mode

+ crossover (metal over diffusion; metal over polysilicon; intersecting vertical and horizontal methods)

* contact (to polysilicon or diffusion)

| polysilicon or diffusion runner

: metal in vertical direction

- metal in horizontal direction

The width of the transistors can be increased by including multiples of the symbols.(E or D).

Stick diagrams allow a high degree of freedom to optimize the layout since they are easy and fast to draw. Area reduction can achieved by manipulating the stick diagram. The circuit performance can also be optimized by varying the transistor sizes based on the results of circuit simulation. This modification can easily done by operating on the symbolic description of the layout. It is only a matter of adding more E's and D's to the coded stick diagram.

3.4 DESIGN OF ARITHMETIC LOGIC UNIT

This is the heart of the data path unit. Logic and arithmetic manipulations on the data are done here. The unit must be designed to function very efficiently, in order to maintain a high overall system performance.

3.4.1 Carry chain circuit

This is the first functional block to be designed since the delay in the carry chain might limit the system performance. We choose to employ the so-called manchester type carry chain [1] shown in Fig. 3.8, for propagating carry signals. In each stage of the adder, a carry propagate signals is derived from the two input variables to the adder, and if it is desired to propagate the carry, this propagate signal is applied to the gate of an enhancement mode pass transistor. The source of the transistor is carry-in and drain is carry out. Thus the carry can be propagated from one end to another without inserting a full inverter delay between stages.

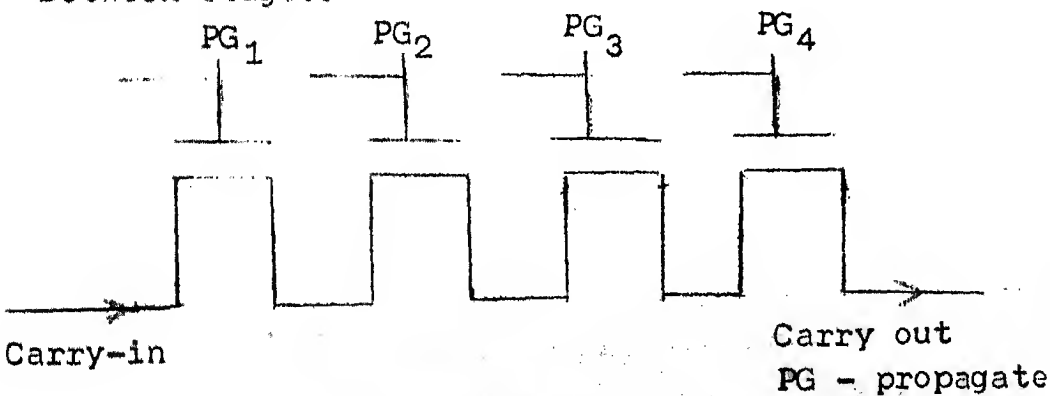


Fig. 3.8

But as the number of stages are increased, the delay will become large. So, it is necessary to group the pass transistors into sections and interpose inverting logic between the sections.

It is observed that the carry chain can propagate a low signal quite fast but its rise transient is rather slow and hence it is slower in propagating a high carry signal. To counter this, we precharge the carry chain during φ_1, when the ALU is idle. In order to improve the performance of the circuit we employ a carry look ahead scheme using a simple carry by pass technique (Fig. 3.9).

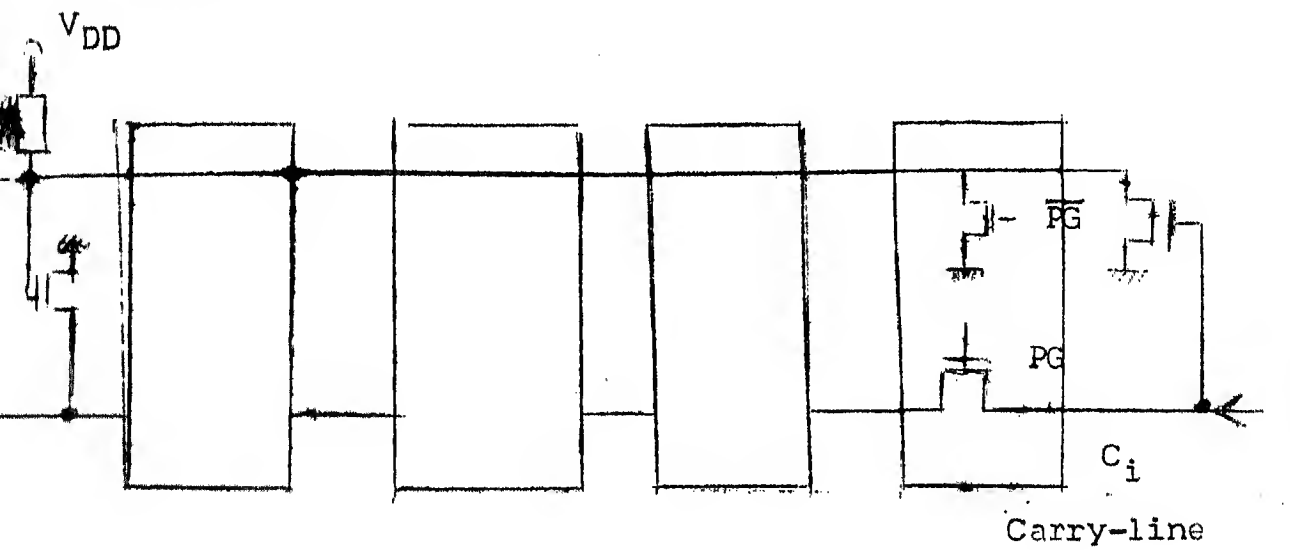


Fig. 3.9

We adopt the bit-slice approach in the ALU design since results in a highly regular design. In each 4 bit block carry is calculated in parallel and then it is rippled

through from block to block. The precharged carry bus and the carry by pass circuit will speed up the operation considerably.

The carry chain circuit is shown in Fig. 3.10.

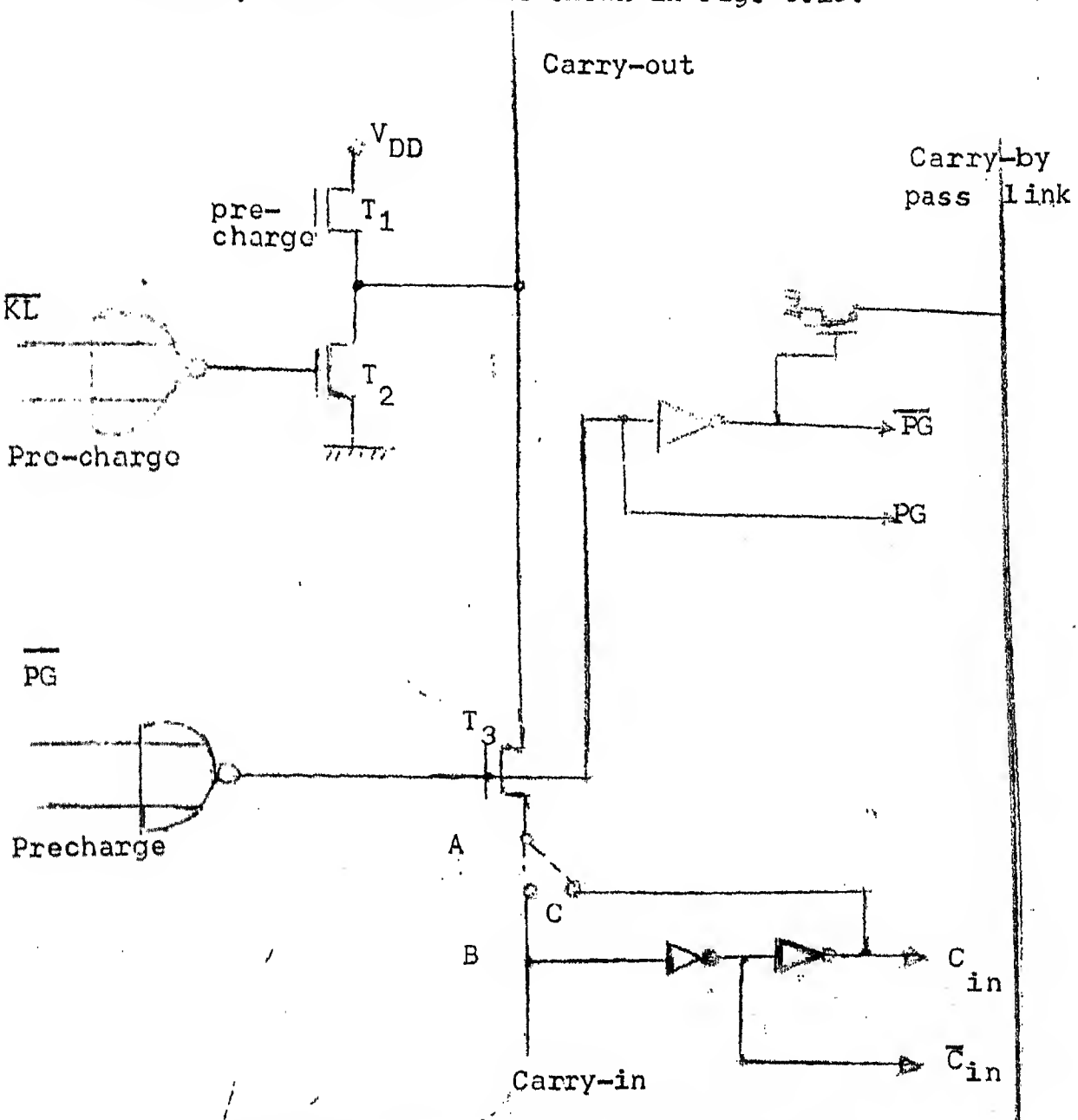


Fig. 3.10

The propogate (PG) and carry kill (KL) signals are generated by two NOR gates which have $\overline{PG}$ and $\overline{KL}$ as one input and the precharge as the second input. So the PG and KL signals are disabled when the precharging takes place during φ_1. The carry chain runs through the pass transistor T_3 from carry into carry out. The carry chain is precharged during φ_1 through the transistor T_1. The carry kill signal, derived from the inputs to the ALU, simply grounds the carry chain through the transistor T_2, if KL is high. The propagate signal, also derived from the ALU inputs will cause carry out to be equal to carry in, if PG is high.

It is observed that all interesting combinations of carry in and the input signals can be generated by using PG, $\overline{PG}$ and C_{in}, $\overline{C}_{in}$ from each stage. We have seen that in order to minimize the propagation delay through the carry chain we have to interpose inverters between sections of pass transistors. We recognize that each carry chain function block contains two inverters that produce carry in at their output having been twice inverted from the actual carry in signal. We can merely substitute this buffered carry in signal for the actual carry in signal to minimize the delay. For a 16 bit processor this can be done at every 4th stage, the connection between A and C is made.

The carry bypass line is also shown in Fig. 3.10.

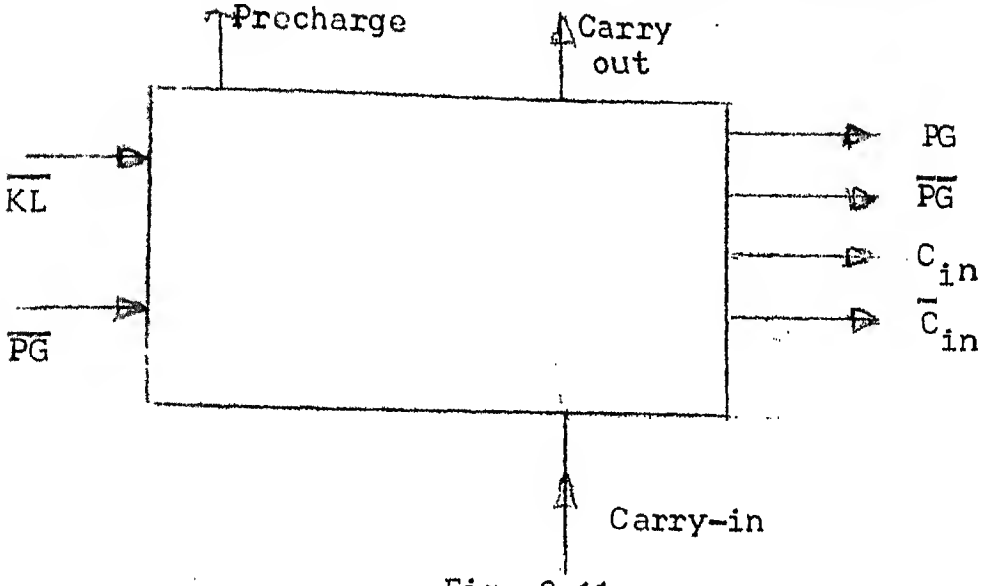


Fig. 3.11

3.4.2 Functional abstraction of the carry chain

The block diagram is shown in Fig. 3.11.

The circuit can be represented as a logic block with two inputs $\overline{KL}$ and $\overline{PG}$, outputs, PG, $\overline{PG}$, C_{in}, $\overline{C}_{in}$; carry in and carry out and one control signal, precharge.

Now, the task is to design functional blocks [1] to combine two input variables to form $\overline{PG}$ and $\overline{KL}$, combine carry in and propagate to form the output, and designing drivers for controlling logic.

Now, the functional blocks to derive $\overline{PG}$, $\overline{KL}$ and output are designed. We go for a highly regular structure with minimum delay, minimum power, and minimum area. ~~The circuit~~

The circuit diagram of this so-called general logic functional block is shown in Fig. 3.12.

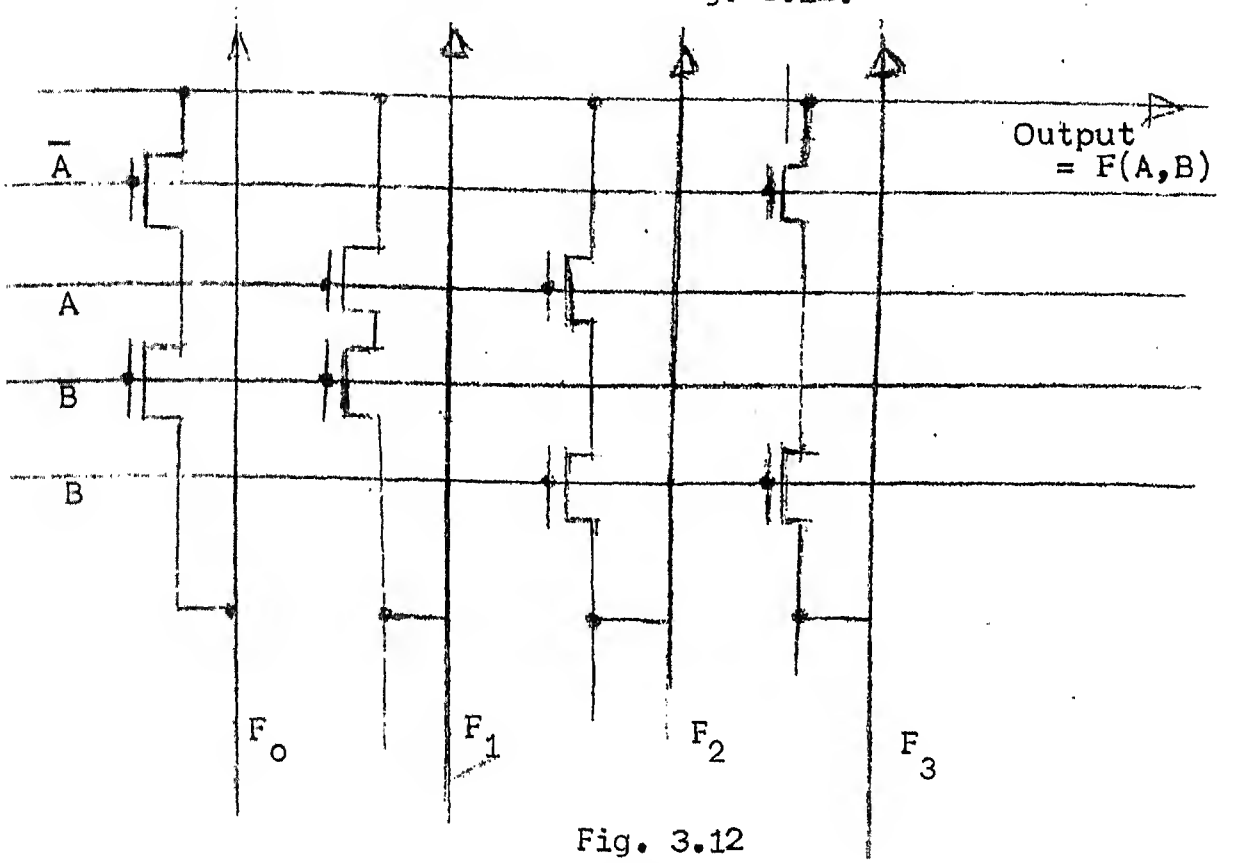


Fig. 3.12

The circuit consists of a set of transistors that fully decode the input combination of A and B. The set connects only one of the vertical control lines to the output, depending on the input combination. For example, if A and B are both high then the control wire F1 is connected to the output. The truth table entries for the desired logic function are placed on the vertical wires, the output is then the desired logic function of the two input variables. For example, if the logical-OR of A and B is required, a logic-0 is applied to F_o, and a logic-1 is

applied to F_1, F_2 and F_3. The control lines (F_o-F_3) need be generated only once and they run through every one of the 16 bit slices.

Functional abstraction of the logic block is as shown in Fig. 3.13.

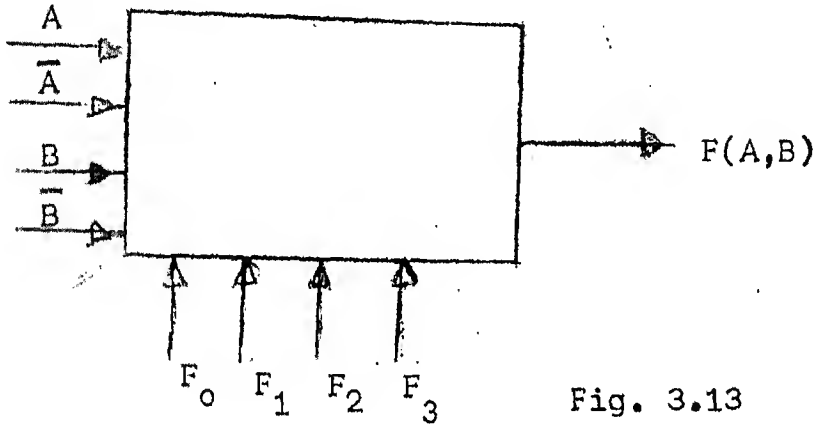


Fig. 3.13

3.4.3 Block diagram of ALU bit slice

The block diagram is shown in Fig. 3.14.

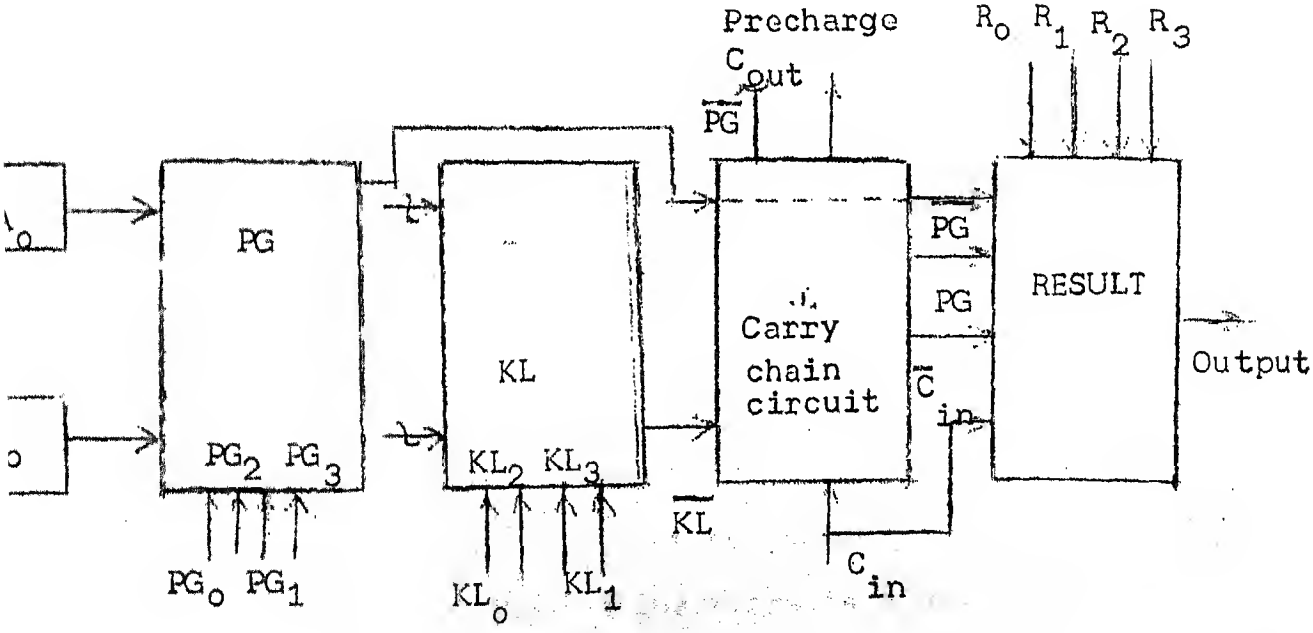


Fig. 3.14

The functional dependence of the output on the two inputs and the state of the carry is determined by PG_0 through PG_3, KL_0 through KL_3 and R_0 through R_3, along with carry in to the least significant bit of the ALU.

The main advantage of this design is that it is very general and the details of its operation can be left unbound until a later time.

The ALU input register and the control circuit for generating the PG, KL and R control lines are designed next.

3.4.4 ALU input register

ALU must have two input registers for storing the two operands. The circuit of a Register bit slice is shown in Fig. 3.15.

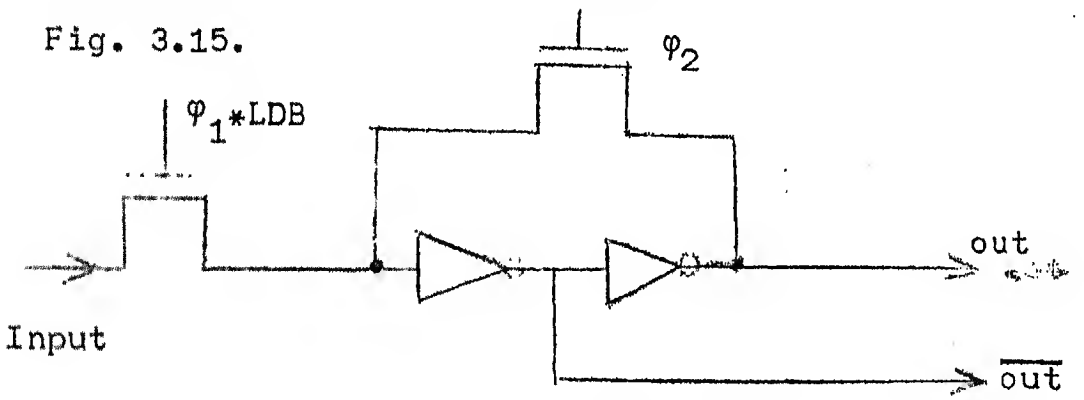


Fig. 3.15

The control signal (φ_1 * LDB) decides whether the bus is to be loaded into the latch or not during φ_1. The feedback transistor around the two inverters is always activated

during φ_2. Thus the data will stay indefinitely, if unchanged. Both the output and its complement are available as required by the functional blocks of the ALU.

3.4.5 ALU control wires

The op-code specifying the state of each control wire arrives during φ_1, when the ALU is being precharged. It must be latched and applied to PG, KL, and R control wires during φ_2. Since PG, KL and R function blocks consist of pass transistors it is necessary to precharge the outputs during φ_1. This is best done by maintaining the PG, KL, and R control wires high during φ_1.

The circuit shown in Fig. 3.16 is suppose to achieve the above purpose.

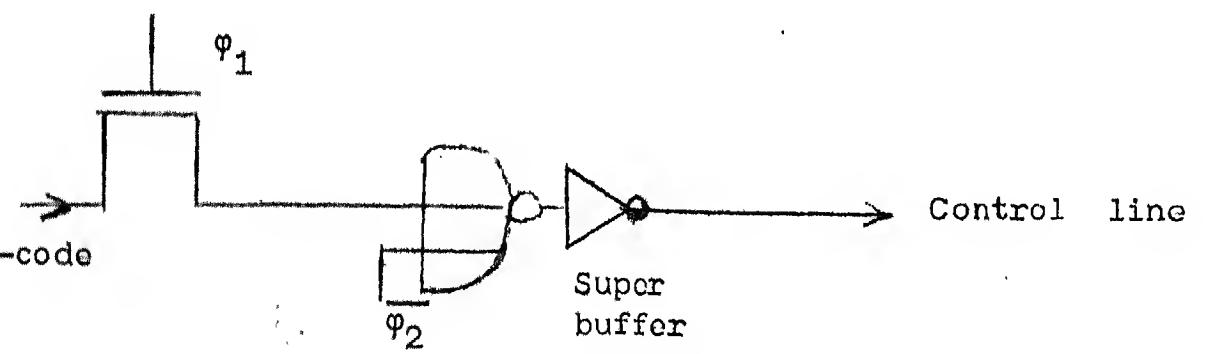


Fig. 3.16

The code is latched during φ_1, and forms one input of the NOR gate. The other input is $\overline{\varphi_2}$, thus the output of the NOR gate is forced to be low during φ_1. The NOR gate output is applied to the gate of an inverting super buffer, so that the output is guaranteed to be high during φ_1. During φ_2, the code is driven onto the PG, KL and R control wires.

3.4.6 HOW does the ALU function?

This is illustrated by taking some examples. The following tables gives the function to be performed by ALU and the required control wires.

Function	KL_0	KL_1	KL_2	KL_3	PG_0	PG_1	PG_2	PG_3	R_0	R_1	R_2	R_3	C_{in}
Add A+B	1	0	0	0	0	1	0	1	0	1	0	1	0
Decrement A-1	0	1	1	0	1	0	0	1	1	0	1	0	1
Logical OR A or B	0	0	0	0	0	1	1	1	0	1	1	0	0

4-bit operands are taken for illustration purposes.

ADD :

A	1	0	1	1	$A_3\ A_2\ A_1\ A_o$
B	1	1	0	1	$B_3\ B_2\ B_1\ B_o$
C_{in}	1	1	1	0	$C_{in3}\ C_{in2}\ C_{in1}\ C_{ino}$
	1	1	0	0	0

C_{ino} is the carry into the least significant bit of ALU

carry out of the most significant bit

Consider the Oth bit position. A_o and B_o both are 1. So, from the function block circuit we see that KL is zero, and PG is also zero. Hence, carry is generated, C_{in1} is 1. Output of the R-function block is zero since both PG, and C_{ino} are zero.

For the 1st bit position :

A_1 is 1, B_1 is 0, so, KL is zero, PG is one, hence C_{in2} is 1.

Output is zero because both PG, C_{in1} are high.

Similarly, we can determine the outputs for each bit position.

It can be verified that by altering the control wire code, the ALU can perform operations, such as, addition, subtraction, negation, complementation, increment, decrement and logical operations, such as, logical OR, EX-OR, AND, etc...

3.5 CIRCUIT DETAILS

Before going to the layout phase, one must have an idea of the circuit details such as the pull up to pull down ratio in an inverter, the threshold voltages of the inverter, pull-up, pull-down, and pass transistor. We should then manipulate these values to achieve the desired performance. This is very helpful in actual circuit design. Circuit representation of the MOS transistor is shown in Fig. 3.17.

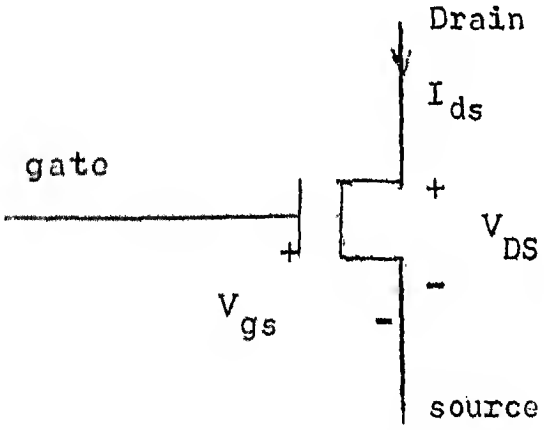


Fig. 3.17

Let 'L' be the length of the channel and 'W' be the width of the channel. Let Vth be the threshold voltage of the enhancement mode transistor. Let us define transit time as the time taken by an electron to move from source to drain. We denote this by 'τ'

$$\tau = L/\text{Velocity} = L/\mu E = L^2/\mu V_{ds}$$

μ mobility

E electric field from source to drain.

We can derive the expression for I_{ds} (source to drain current) [1]

$$I_{ds} = \frac{\mu \epsilon W}{LD} (V_{gs} - V_{th}) V_{ds} \qquad (1)$$

where ϵ is the permittivity of the insulating material (between the gate and the channel) and D is the thickness of the same.

The gate to channel capacitance C_g is given by

$$C_g = \epsilon WL/D$$

If Vd_s is low, the MOS transistor can be modelled as a resistor whose resistance is given by

$$R = \frac{V_{ds}}{I_{ds}} = \frac{L^2}{\mu C_g (V_{gs} - V_{th})} \qquad (2)$$

or the time constant

$$RC_g = \frac{L^2}{\mu (V_{gs} - V_{th})} \qquad (3)$$

These simple equations will help us make useful design decisions. The transit time τ is the minimum time in which a charge placed on the gate of one transistor results in the transfer of a similar charge through that transistor's channel on to the gate of a subsequent one. Thus τ can be viewed as a basic unit of time in an integrated system.

While in saturation

$$I_{ds} = \frac{\mu\varepsilon W}{2LD} (V_{gs}-V_{th})^2 \qquad (4)$$

Let

V_{inv} be the threshold voltage of the inverter

As V_{in} rises above V_{inv} : V_{out} approaches.

Let the Z_{pu} be the length to width ratio of pull-up, and Z_{pd} be that of pull-down, i.e., $Z_{pu} = L_{pu}/W_{pu}$ and $Z_{pd} = L_{pd}/W_{pd}$

Then, it can be shown that [1]

$$V_{inv} = V_{th} - \frac{V_{dep}}{\sqrt{Z_{pu}/Z_{pd}}}$$

where V_{dep} is the threshold voltage of the pull-up.

From eqn. (4), lesser the V_{th}, more the current driving capability of the pull-down. But if V_{th} is too low, the inverter outputs will not be able to turn off pass transistor used as simple switches. For a 5-V supply V_{th} could be between 0.75V - 1V.

As V_{dep} is made more negative, the current driving capability of the pull-up increases. But for a given V_{inv} and V_{th}, if we increase V_{dep} (make it more negative), then we have to increase the pull-up area. For normal inverters we may choose V_{dep} so that with gate tied to the source, they turn on approximately as fast as a pull-down with V_{DD} connected to gate and source grounded. For a 5-V supply V_{dep} can be between -3V to -3.5V.

It is desirable to have $V_{inv} = 0.5\ V_{DD}$ and hence Z_{pu}/Z_{pd} should be about 4:1.

In the case of pull-ups used for static dual port registers, we may choose the threshold voltage (V_{dep}) to be about 1.5V. This provides low supply currents (can be seen from the above equations). V_{inv}, in this case won't be exactly midway between V_{DD} and GND.

NMOS logic is a ratio type logic and so the pull-up has lees driving capability than the pull-down. In order to offset this, we use super buffers to drive capacitive loads. One such buffer is used in the ALU control driver, mentioned previously [1].

Figure is shown in Fig. 3.18.

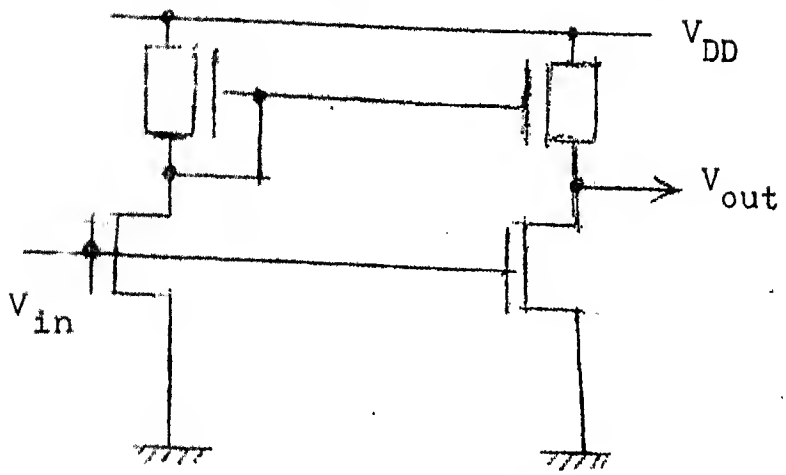


Fig. 3.18

Here the gate of the pull-up is connected to a signal that is the complement of that driving the pull down. When the pull-down transistor gate is at a high voltage, the pull-up gate will be low, and the current through the super buffer will be the same as that of a standard inverter of same size. But when the gate of the pull-down is driven to zero, the pull-up will get approximately twice the drive as it is the only load on the output of the previous inverter. Since the gate drive is twice the normal, the current sourcing capabil is four times that of a standard inverter. Hence, the curr sourcing capability of the pull-ups is same as the current sinking capability of pull downs.

One more point is that when we use pass transistors between inverter stages, the ratio Z_{pu}/Z_{pd} must be 8:1

for that inverter whose gate is connected to one end of the pass transistor in order to maintain uniform output voltages. [1]. We differentiate this in the stick diagram layout by putting a cross in the box meant for the depletion mode transistor.

3.6 LAYOUT OF THE ALU

In this thesis, the layout of ALU bit slice, and the layout of register and shifter unit are shown. Layout strategy for the ALU bit slice differs slightly from the 'GATE matrix layout' method described earlier. Registers and the shifter unit layout is done using the 'gate matrix layout' matrix. This is done due to two reasons :

1. The circuit design of the ALU is such that a slightly different layout strategy results in a better density, better performance layout.
2. It is desirable to see the difference between the well structured and regular layout and a slightly unstructured one.

The stick diagram of the ALU and the input registers is shown in Figs.3.19 and 3.20 .

The ALU layout is slightly different from the 'gate matrix layout' method. PG, KL and R control lines should run through the ALU, vertically. We can't run them on poly because they don't control the gates of pass transistors. So, we

adopt the following strategy : PG, KL and R control lines run on metal vertically. The clock and power busses also run on metal vertically. The polysilicon lines run in the horizontal direction. Other control lines run vertically on metal. The stick figure of the carry chain is shown in Fig. 3.19. Stick diagram of the function block is shown in Fig. 3.21.

CHAPTER 4

DESIGN OF SUBSYSTEMS

4.1 DESIGN AND LAYOUT OF REGISTER FILE

A set of 16 Registers are incorporated in the data path chip. Some of these may be used as general purpose and the rest as special purpose. Since the ALU generally needs two operands, it is convenient to have dual port registers. A typical register cell is shown in Fig. 4.1.

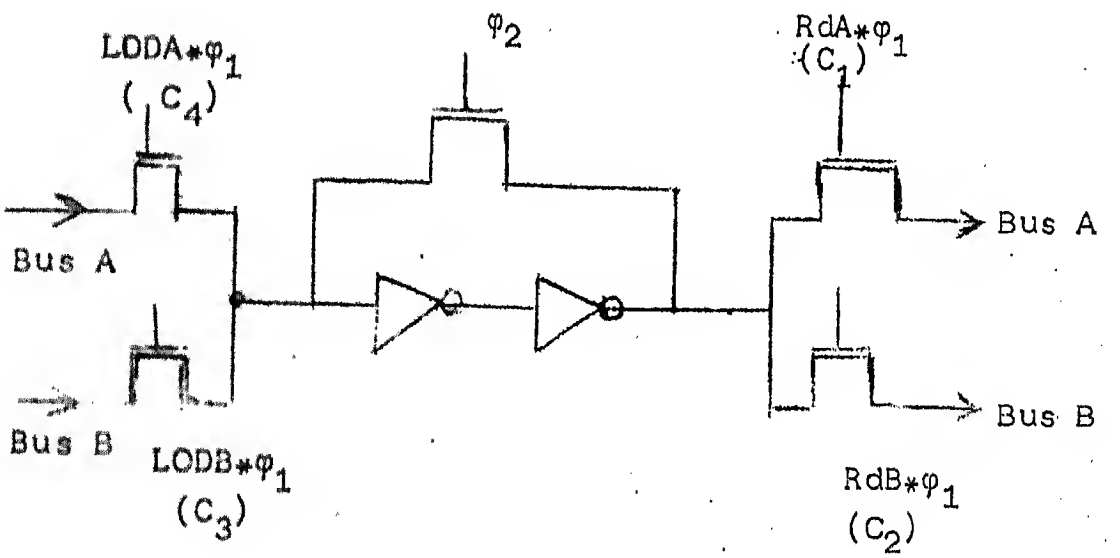


Fig. 4.1

It is a static register with input multiplexer and output drivers.

Data may be loaded into the register cell from Bus A, via T_4, by driving C_4 high. Register cell can also be

loaded from Bus B, via T_3, by driving C_3 high. Data can be read by bus A, via T_1, if C_1 is high. Bus B can read the register output, via T_2, if C_2 is high. The functional abstraction of the register cell is shown in Fig. 4.2.

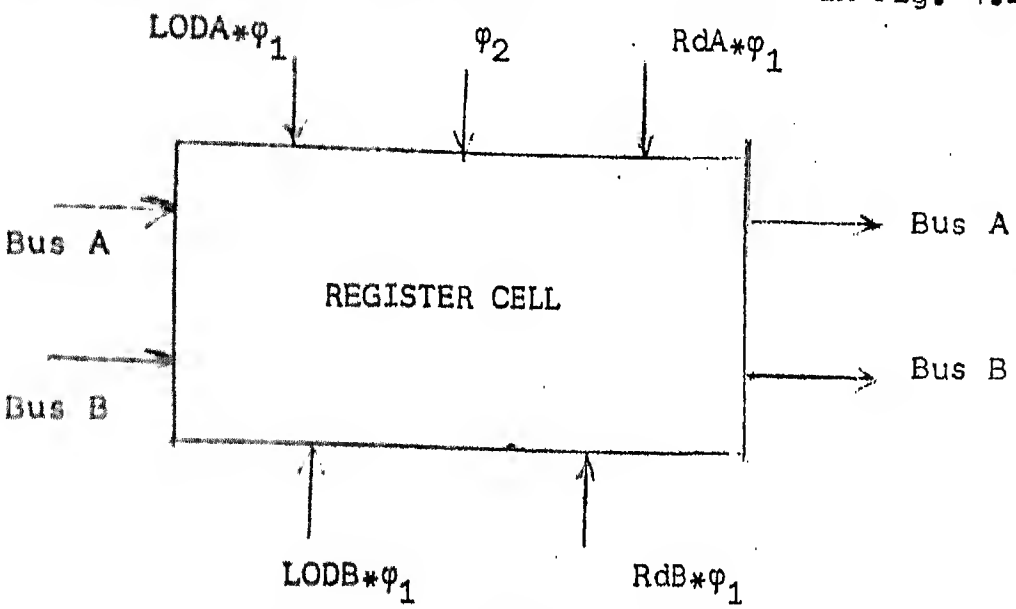


Fig. 4.2

The stick figure of the register cell is shown in Fig. 4.3. The digitized version is shown in Fig. 4.4.

4.2 DESIGN OF SHIFTER/TEMPORARY REGISTER

If a 16 bit processor is to handle variable length words, it requires a shifter of at least 16-bits long. The shifter is useful in field extractions, instruction decoding, assembly of bit maps for graphics. It may also be used in packing bytes into a word and for unpacking a word. Both the buses run through the shifter unit and the shifter can thus select any continuous segment of 16-bits from the 32-bit word, [1], formed by concatenating the two buses.

The temporary register is to latch the shifter output.

The circuit diagram of the shifter is shown in Fig.4.5. The shifter outputs are precharged during φ_2. Hence, the pass transistors forming the shifter array are required to pull down the shifter outputs only when the appropriate bus is pulled low. The shift constant [0:15] control the gates of the pass transistors, which connect the buses to the appropriate outputs. Note that, only one of the shift signals [0:15] is high during the period the shift is occurring.

The operation of the shifter is illustrated by taking a 4 by 4 bit shifter, shown in Fig. 4.6. The shift constant specifies the number of bits from bus B present in the output.

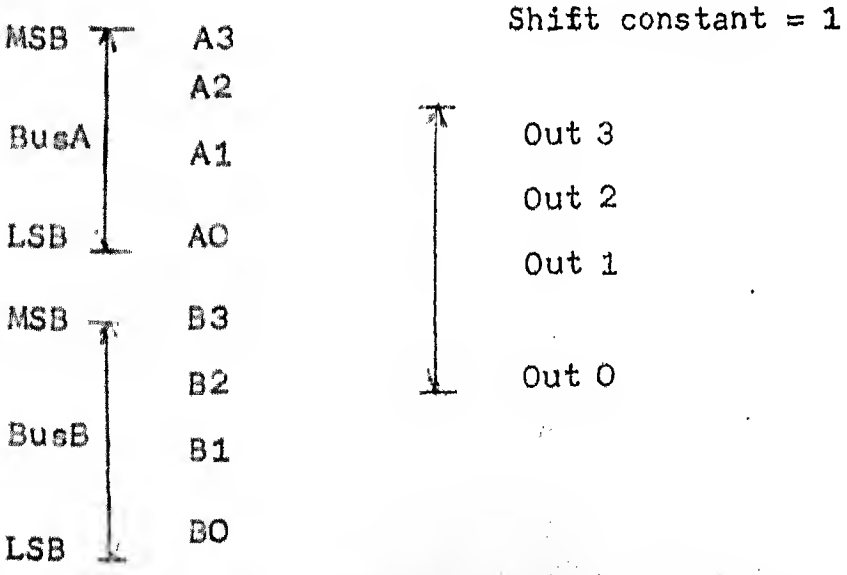


Fig. 4.6

Shift constant 0 returns the A bus, shift constant 1 returns the MSB of bus B in the LSB of output and the remaining bits from Bus A as shown in Fig. 4. 6. So the shift constant determines the bit position where the output window starts.

Note that the shifting function is accomplished in a single clock period [during φ_1] and the circuit does not dissipate any static power. The block diagram abstraction is shown in Fig. 4.7.

Temporary

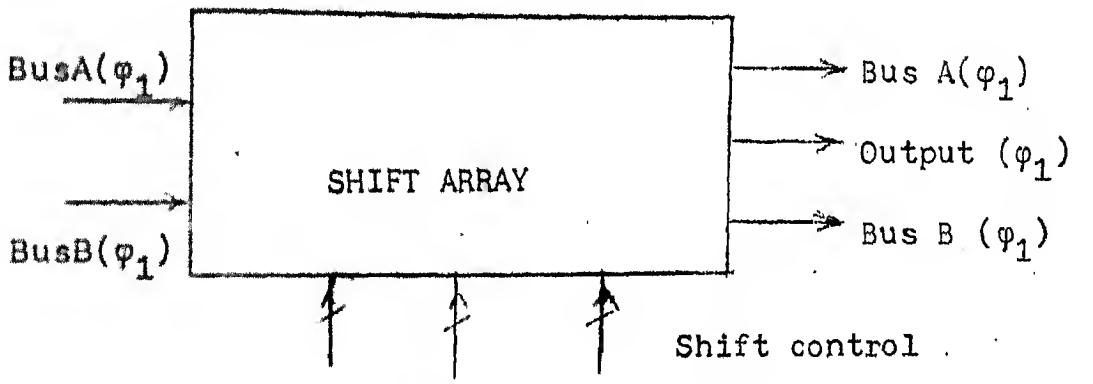


Fig. 4.7

Temporary Register :

The circuit diagram is shown in Fig. 4.8. The register should latch in the shifter output during φ_1. It's output is accessed by both the buses.

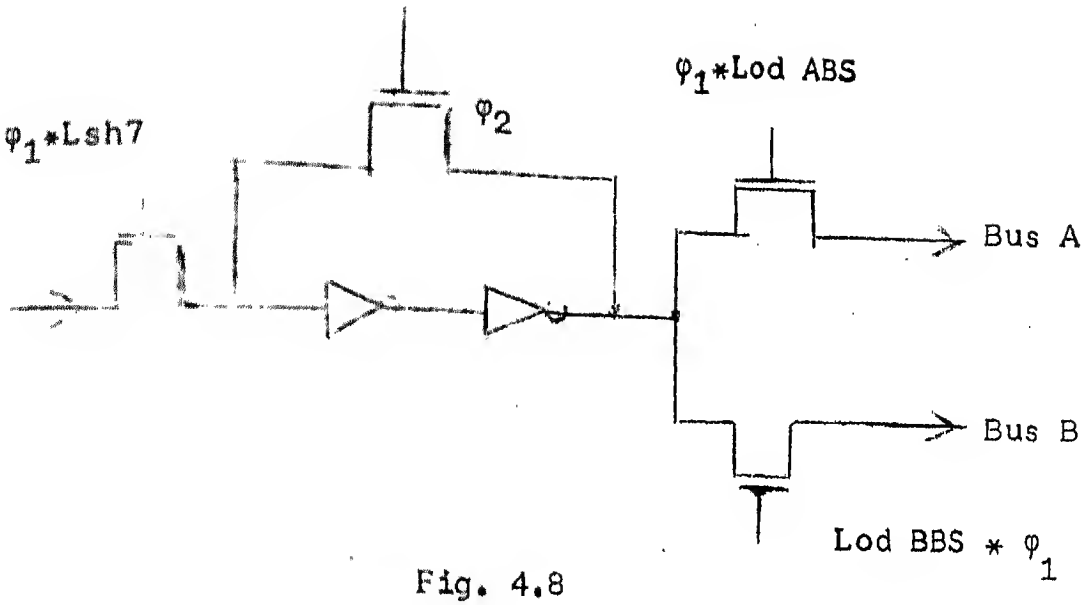


Fig. 4.8

The block diagram of the temporary register is shown in Fig. 4.9. The stick diagram of the shifter unit is shown in Fig. 4.10. The digitized stick figure is shown in Fig.4.11.

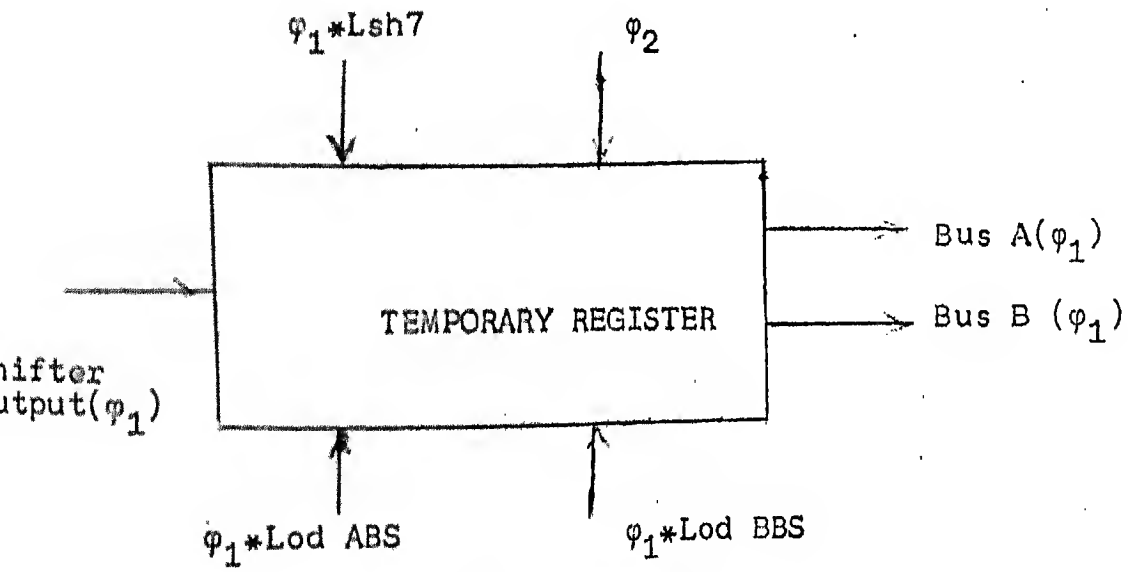


Fig. 4.9

4.3 DESIGN OF ALU-OUTPUT REGISTER/SHIFTER

The major function of the ALU-output register is to latch the ALU output at the end of φ_2. In order to support the multiplication operation, the register should be 32-bits long, and it must work as a shift register with the right shifting capability.

The timing signals for latching the ALU output, and for shifting the data are to be derived first (see Fig. 4.12). The ALU operates during φ_2, and the results are assumed to be ready by the end of φ_2 AND CTRL1, denoted by (φ_2 * CTRL1), and they are latched into the ALU output register. The output register performs the shifting function, if required, during φ_2 * CTRL2. In fact, the 32-bit ALU-output register can be viewed as two 16-bit registers connected as shown in Fig. 4.13.

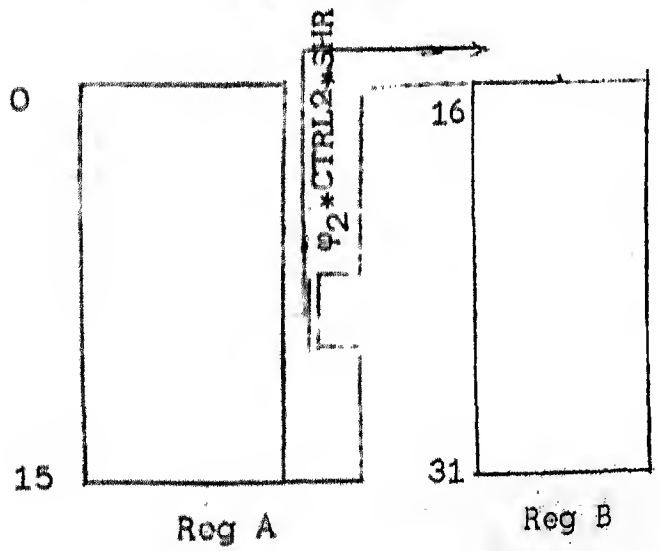


Fig. 4.13

The two registers, each 16 bits wide are concatenated for shifting operation via a pass transistor whose gate is controlled by the signal φ_2 * CTRL2 * SHR (SHR → shift right). The registers are static in nature. The registers must satisfy the following constraints :

1) Both the internal buses must have access to each of the 16-bit registers.

2) ALU output must have the flexibility to be placed in any of the two registers.

The circuit diagram of a register cell satisfying the above discussed constraints is shown in Fig. 4.14.

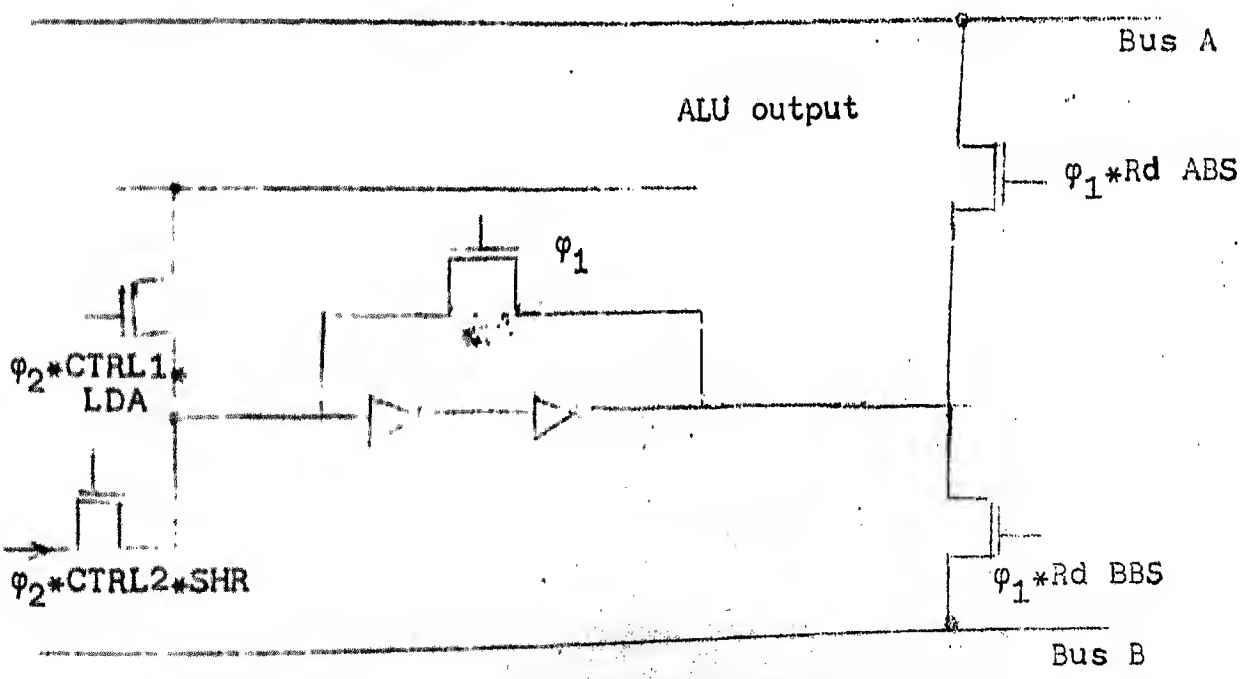


Fig. 4.14

4.4 CONTROL CIRCUIT FOR MULTIPLICATION

In this section a special control circuit for multiplication is described. Before going to the detailed design, let us see how useful it is to have a special control circuit for multiplication.

We are going to employ the two bit booth's algorithm for multiplication of 2's complement numbers. So depending on the previous bit and the current bit of the multiplier the ALU may perform one of the following operations :

1) Add the multiplicand to the partial sum
2) Subtract the multiplicand from the partial sum,
3) Pass the partial sum straight through, without any modification.

One way to do this operation would be to send both the bits to the controller, and execute a conditional branch to one of the three locations. However, it would take several cycles to perform each step of multiplication, if this method is used. So, it is desirable to have a control circu on the data path chip itself to modify the ALU operations depending on the two flag bits. Thus, a single cycle will perform this part of the multiply step.

Fig. 4.15 shows the arrangement of the ALU, ALU-shifter the MULTIPLY control circuit, the microcode decoder and the drivers. ALU input latches are denoted by Latch 1 and

Latch 2. The two 16-bit registers of the ALU-shifter are denoted by ACC1 and ACC2.

If the instruction is anything other than the MULTIPLY instruction, the control lines of the ALU are not disturbed by the MULTIPLY control circuit. If the instruction is a MULTIPLY instruction, the control lines are changed, according to the two flag bits ($ACC2_T$, $ACC2_{T-1}$). Since the opcode for the ALU arrives during φ_1, the opcode is modified according to the flag bits, latched into the control drivers (discussed in ALU design), and made active during φ_2.

The booth's algorithm for multiplication of two n-bit numbers, in the two's complement form, is given below.

Let,

x denote the multiplier in the 2's complement form

y denote the multiplicand in the 2's complement

PSUM denote the partial sum.

Initially PSUM = 0

$$x_{T-1} = 0$$

In each cycle the multiplier along with the PSUM is shifted one place to the right.

The booth's algorithm is illustrated in Fig. 4.16.

x_{T-1}	x_T	ALU operation
0	0	R = PSUM
0	1	R = PSUM-Y
1	0	R = PSUM+Y
1	1	R = PSUM

Fig. 4.16

The multiplication operation is carried out as described below. The register organization is as shown in Fig. 4.17.

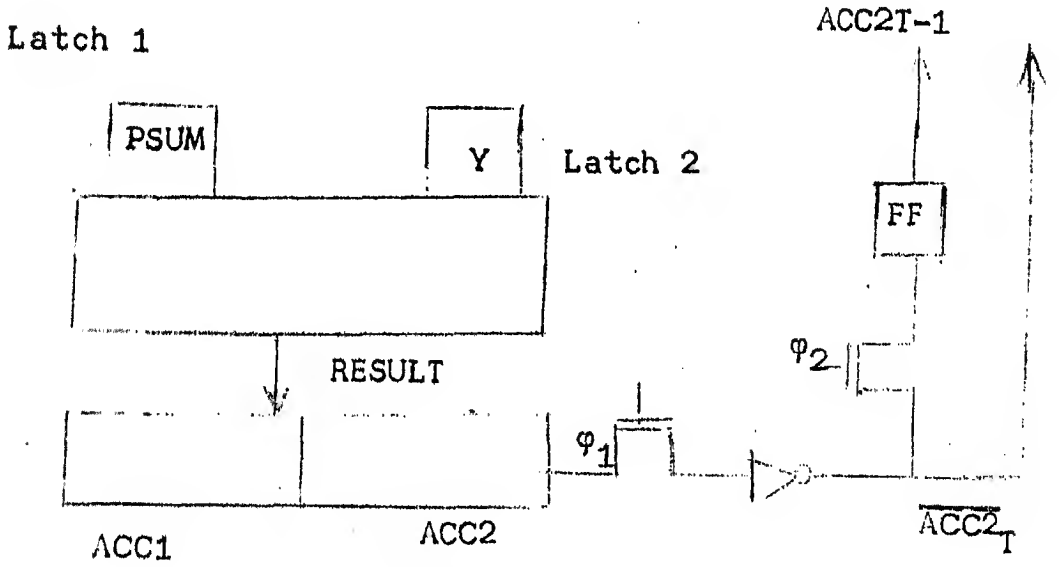


Fig. 4.17

To start with, ACC2 has the multiplier x, the latch 2 has the multiplicand Y, ACC1 and latch 1 has zeros. FF (flip-flop) hich holds $ACC2_{T-1}$ is cleared.

During the first φ_2, the ALU will perform either addition, subtraction or push PSUM through depending on the LSB of x, and FF which is cleared. The result R is latched into ACC1 by the end of φ_2 * CTRL1.

Note that during φ_2, the FF would have acquired the value of the LSB of ACC2. The right shift occurs during φ_2 * CTRL2. Now, the new LSB of ACC2 will become $ACC2_T$. $ACC2_T$ and $ACC2_{T-1}$ will change the ALU control lines during φ_1, before they are latched in. During φ_1, the PSUM which is in ACC1 is transferred to the latch 1. The next cycle then repeats.

Now, we draw the truth table for the multiply control circuit, giving the corresponding control line values. The truth table is consistent with the functional block circuit diagram discussed in Chapter 3.(see Fig.4.27 for Truth Table).

The circuit uses a small amount of random logic, without seriously effecting the performance. Each control signal has two lines. Depending on the control signals P, and Q aone of them is connected to the input of the latch/driver.(Figs. 4.18, and 4.19).

4.5 BUS INTERFACE (CONTROL) UNIT

We have two internal buses, which run through the data path chip to connect all the functional blocks of the system. We did not use tri-state drivers to source data on to the internal buses. We simply made use of the pass transistors

switches to make sure that no two sources are given control of the bus at the same time. Since the internal buses are idle during the φ_2 period, we choose to precharge the buses during the φ_2 period. Thus, the pass transistors need only pull the bus down, if necessary. We save lot of space by not using tri-state drivers.

The bus interface unit must perform two major functions :

1. It must multiplex the two internal buses and drive the data of the chip into the system bus (both synchronous and asynchronous)
2. It should latch the data (asynchronously or synchronously) and then put the data on to one of the internal buses.

Hence, we classify the bus interface unit into two subunits. One is the 'system bus driver' unit and the other 'system bus receiver' unit.

4.5.1 System bus driver

The requirements are :

1. It must latch the information from one of the buses during φ_1, and should have the capability to retain the same for a number of clock cycles, if necessary.
2. The latch must connect to a bonding pad via a tri-state driver, with an external enable.

To satisfy the first requirement, we use the static register shown in Fig. 4.20.

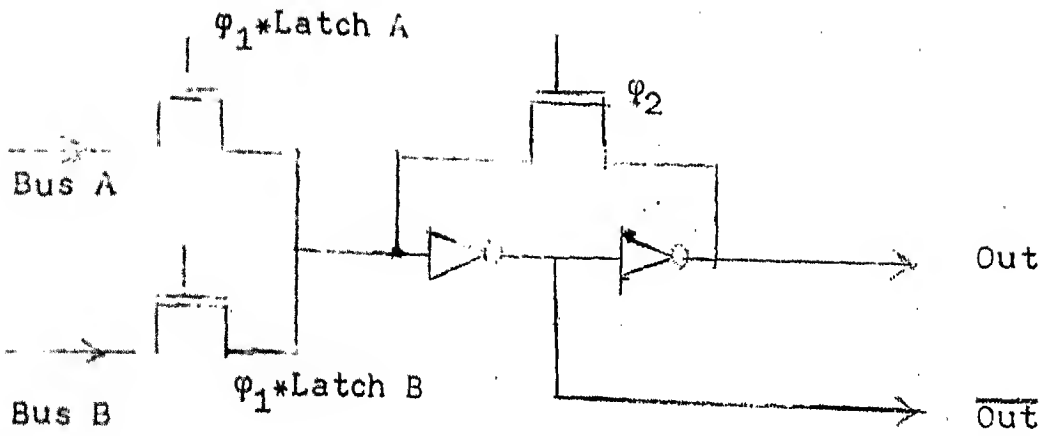


Fig. 4.20

The circuit satisfies requirement No.1. Both $\overline{out}$ and out are available, if needed. One point to note is that the inverters, used in the latch are larger in size than the typical inverters used elsewhere in the circuit. This technique will help in minimizing the delay in driving the data off the chip.

The tri-state driver is controlled by an external pin. We assume that when the control pin is pulled low, the tri-state drivers are enabled and the data is transferred asynchronously. When the control input is high, the drivers are disabled and they are in a high impedance state. However, for synchronous transfer, the data is driven off chip during φ_1 itself. For synchronous transfer, control pin is left floating.

The tri-state driver consists of two buffer stages followed by a pad driver stage (see Fig. 4.21).

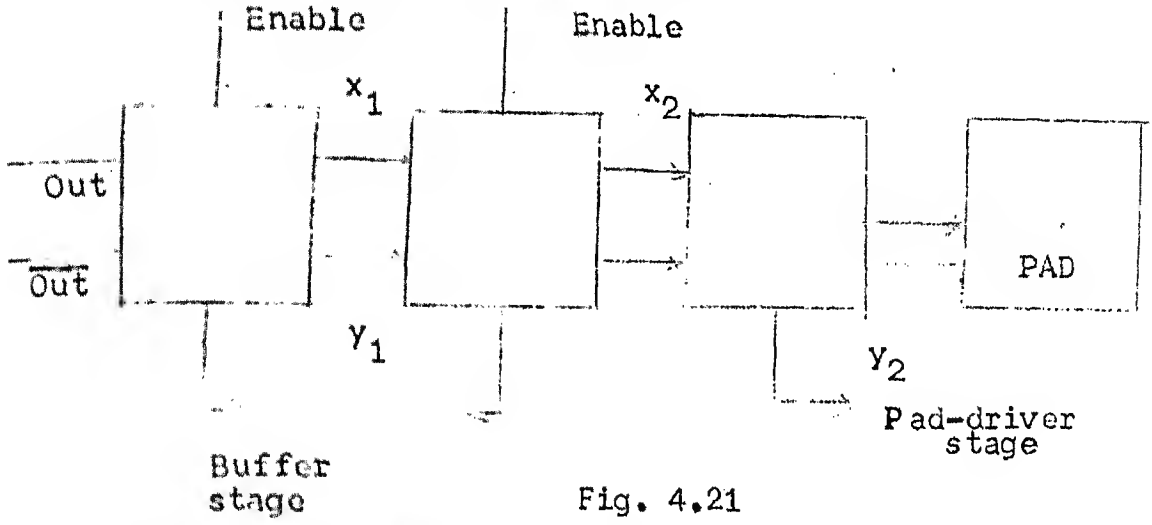


Fig. 4.21

Pad-driver stage :

This consists of two enhancement mode transistors, connected in series. The gates of these transistors are driven by two signals from the last buffer stage. If the enable pin is low, one signal is the complement of the other. If not, both the signals are low. When both the signals are low, the output of the pad driver stage is in a high impedance mode.

The circuit diagram of the pad driver stage is shown in Fig. 4.22.

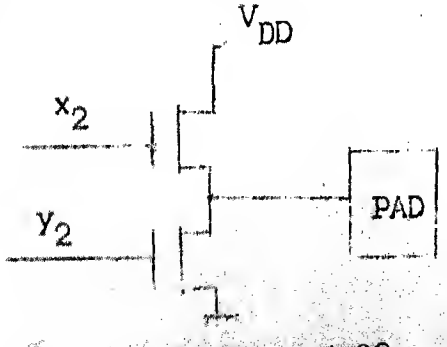


Fig. 4.22

The two enhancement mode transistors have a large W/L ratio, and hence, good current driving capability.

Buffer stage :

The circuit diagram of a buffer stage is shown in Fig. 4.23.

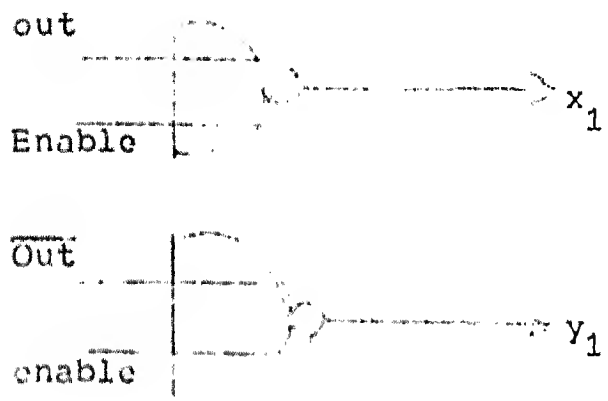


Fig. 4.23

When enable is low, $x_1 = \overline{out}$; y_1 = out.

With two such buffer stages, when the enable pin is pulled low, x_2 and y_2 will be out and $\overline{out}$ respectively. When enable is high, x_2 and y_2 will be low.

4.5.2 System bus receiver

The circuit must have the following requirements :

1. It must be able to latch data from the system bus asynchronously under the control of an external pin (when the external pin is pulled low, the data is latched in).

2) It should be possible to transfer information from system bus to one of the internal buses during the same φ_1 (when this is done, the external control pin can be left floating).

A circuit satisfies the above requirements is shown in Fig. 4.24.

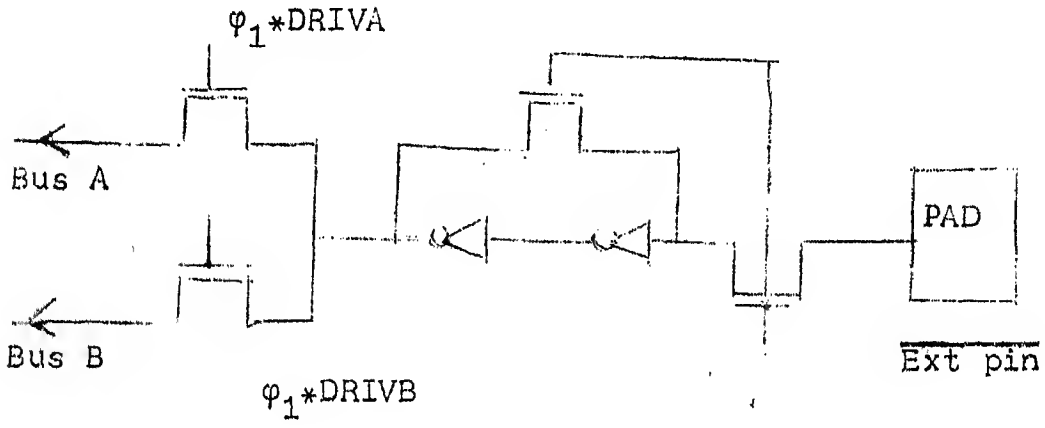


Fig. 4.24

Since the data is latched in asynchronously, the gate of t feedback transistor of the latch is connected to a signal which is the complement of the signal which controls the gate of the input pass transistor of the latch.

4.6 STATUS REGISTER

The four status flags are : carry, zero, overflow, and sign.

The carry flag is the carry out of the MSB of the ALU. The sign flag is the MSB of the ALU. The overflow flag is EX-OR of the two most significant bits of the ALU. The zer

flag is obtained from the ALU output as shown in Fig. 4.25

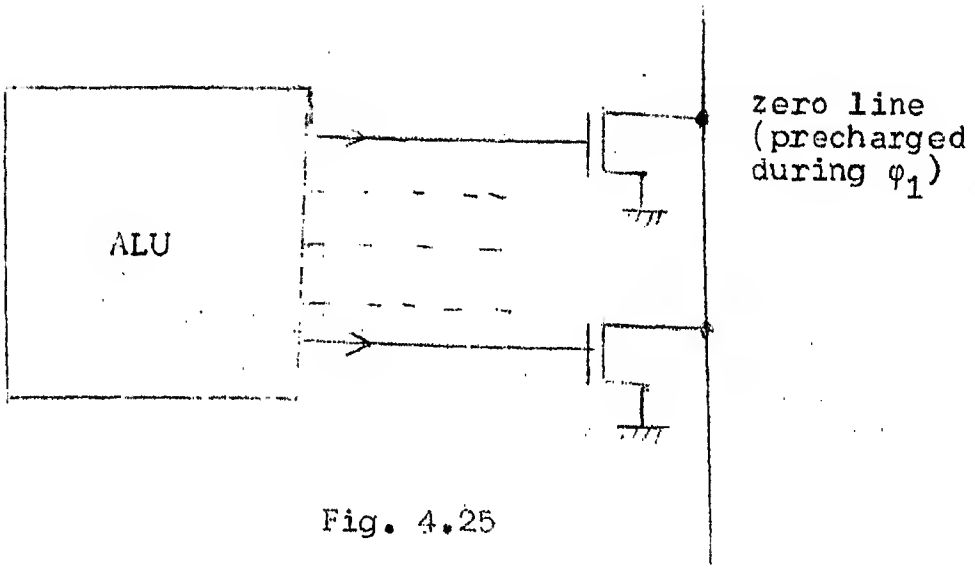


Fig. 4.25

Note that if any of the output bits is high, the zero line is grounded via the pass transistor. As shown, the zero li is precharged during φ_1.

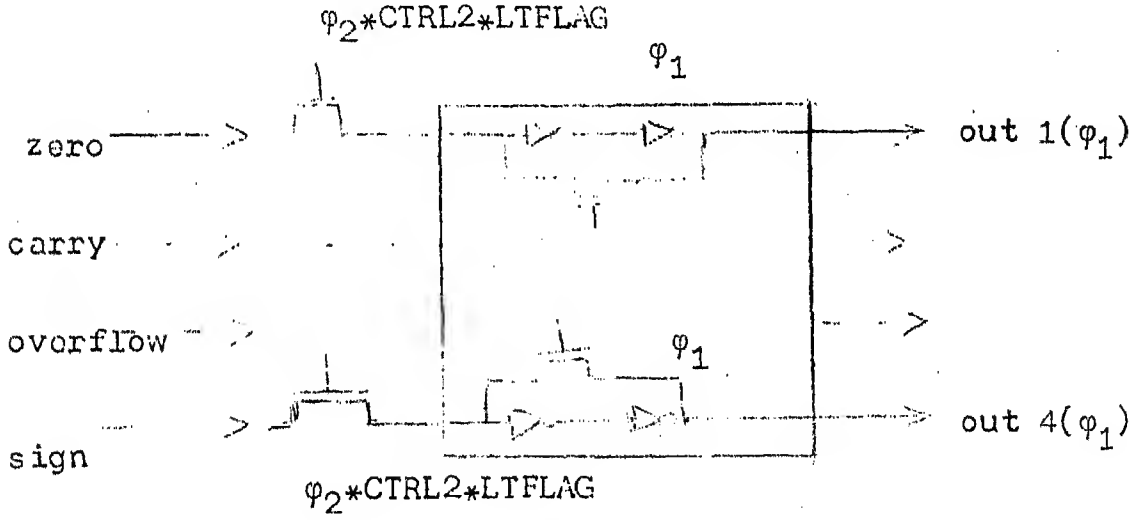


Fig. 4.26

The status register is shown in Fig. 4.26.

The flag bits are latched in during φ_2 * CTRL2, If the LTFLAG signal is high. The flag bits are available at the output during φ_1, for status monitoring.

CHAPTER 5

CONCLUSIONS

A new design approach for LSI/VLSI design is studied. The design strategy is illustated by taking the data path design as an example. The layout of the basic cells of some subsystems is shown. However, the logic needed for generating control signals for the data path unit is not shown.

At the LSI/VLSI level, one is faced with the challenge of designing integrated systems, rather than integrated circuits. Mead and Conway [1] have optimized the design process by combining the concepts of fabrication at the device level and the architecture at the system level to produce truly integrated systems. In the area of structured IC design methodology, they have developed a common design culture, which is very essential in the VLSI area.

The basic concept is simple. It says that integrated circuits are so complex and dense that human designers can not deal with individual devices. Instead, they must be handled at a high level of integrated system architecture. We make use of abstraction to constrain the design complexity. We also make the designs structured and regular, so that only a small number of basic cells are replicated many times

to produce complex systems. Most of the system design is done with the help of logic diagrams, block diagrams, circuit diagrams, and stick figures. This is done in a metric-free topological domain. Thus, at this level, one need not bother about the geometrical details. This helps the designer to concentrate on the topological aspect of the system design. The main criteria is to minimize the area, delay, and power. As far as possible, random logic is not used, thereby improving regularity and minimizing area and power.

If one attempts to design an LSI circuit with a set of small discrete IC's, the result is a design, which is not sound as far as area, delay, and power are concerned.

We try to bypass boolean logic gates as an intermediate step in the design. They are replaced with simple field effect transistor switches and inverters. The design rules are normalized, so that one need not be concerned with the fabrication details of a particular fabrication firm. The time delay can be approximately calculated in terms of the fundamental time unit 'τ'.

Thus, one must aim at producing designs that are simple and efficient as far as time delay, silicon area, and power dissipation are concerned.

SCOPE FOR FUTURE WORK

In this thesis, the circuit design and the layout of some of the basic cells are described. To make an IC chip, complete layout, with the geometric details must be availabl Computer aided design tools will be of great help in obtaini a detailed layout from the initial stick diagrams. Once the stick figures are available, one can use a software package to digitize the stick figure, The digitized figure can be converted to the CIF (Caltech intermediate form) by another software package. The final layout, with the process dependent geometries, can be generated by another program, with the help of the given design rules (geometric information).

So, there is need to develop these software routines, which will aid the designer in generating a detailed layout, from which masks can be obtained. It is also necessary to spread the design culture among the students, mainly to let them know that LSI design is not beyond them. An understanding between the industry (i.e. the fabrication fir and the educational institutions is necessary to make IC design, a purposeful job.

REFERENCES

[1] C. Mead and L. Conway, Introduction to VLSI systems, Addision Wesley, 1980.

[2] Rowson, Lang, Gray, 'A structured design methodology and associated software tools', IEEE Trans. on Ckts and Sys., vol. CAS-28, No.7, July 1981.

[3] 'A hiearchical design rule checking algorithm', Lambda Magazine, vol.2, No.1, 1981.

[4] Marvin E. Daniel and C.W. Gwyn, 'CAD systems for IC design', IEEE Trans. on CAD of Integrated Circuits and Systems, vol. CAD-1, No.1, Jan. 1982.

[5] J.M. Acken and J.D. Stauffer, 'Logic circuit simulatio IEEE Trans. Circuits and Syst., vol.1, No.2, pp 3-12, June 1979.

[6] L.W. Nigel and D.O. Pederson, 'Simulation program with integrated circuit emphasis', in Proc. 16th Midwe Symps. Circuit Theory, April 1973.

[7] B.R. Chawla, H.K. Gummel and P. Kozak, 'MOTIS - An MOS timing simulator', IEEE Trans. Circuits and Syst., vol. CAS-22, pp. 901-909, Dec. 1975.

[8] J.D. Williams, 'Sticks - A new approach to LSI design', MSEE Thesis, MIT, 1977.

[9] A.D. Lopez and H.S. Law, 'A dense gate matrix layout method for MOS VLSI', IEEE J. Solid State Circuits, vol. SC-15, pp. 736-740, Aug. 1980.

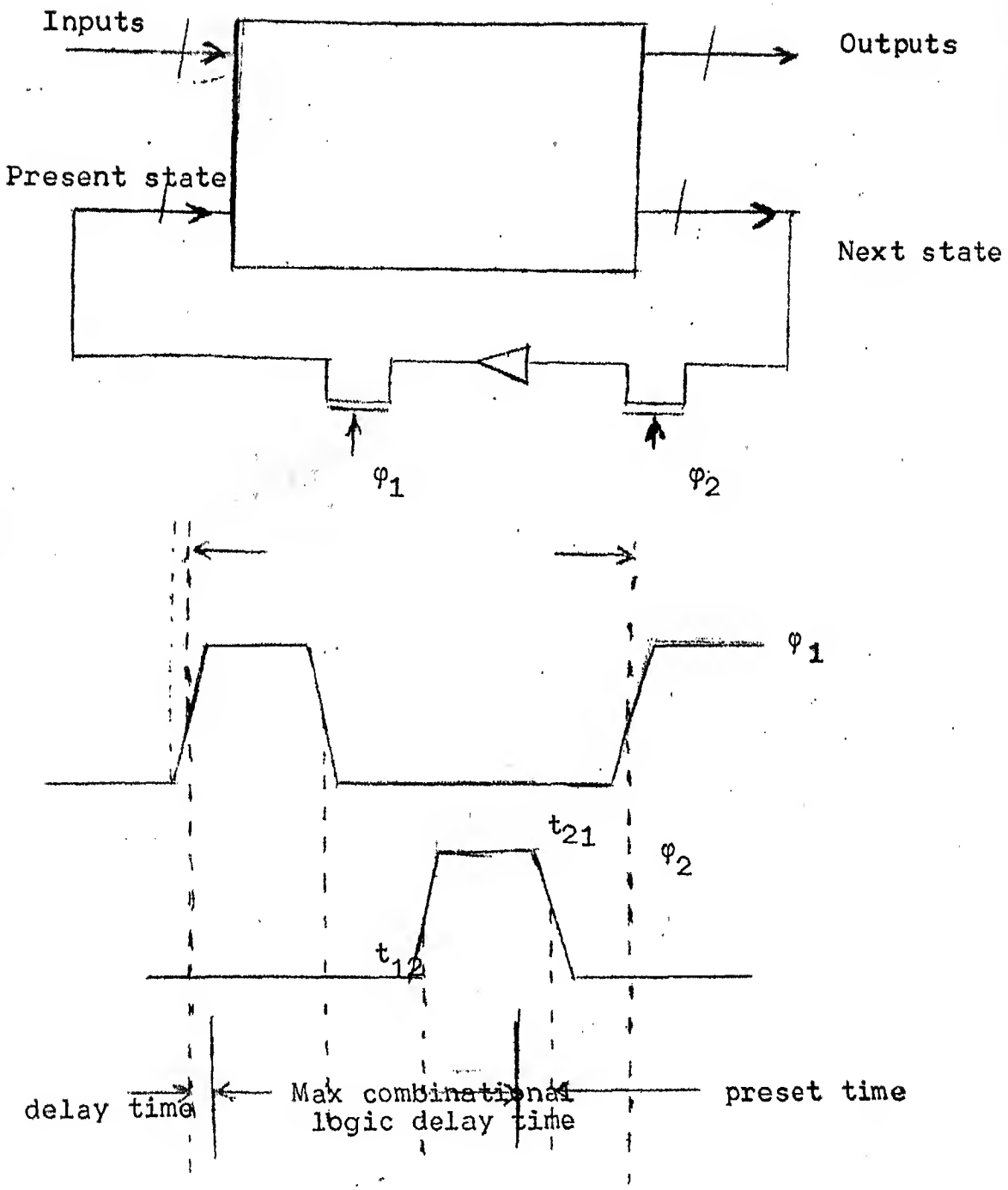

Inputs
Outputs
Present state
Next state
φ_1
φ_2
φ_1
t_{21}
φ_2
t_{12}
delay time
Max combinational logic delay time
preset time
Clock period

Fig. 3.6

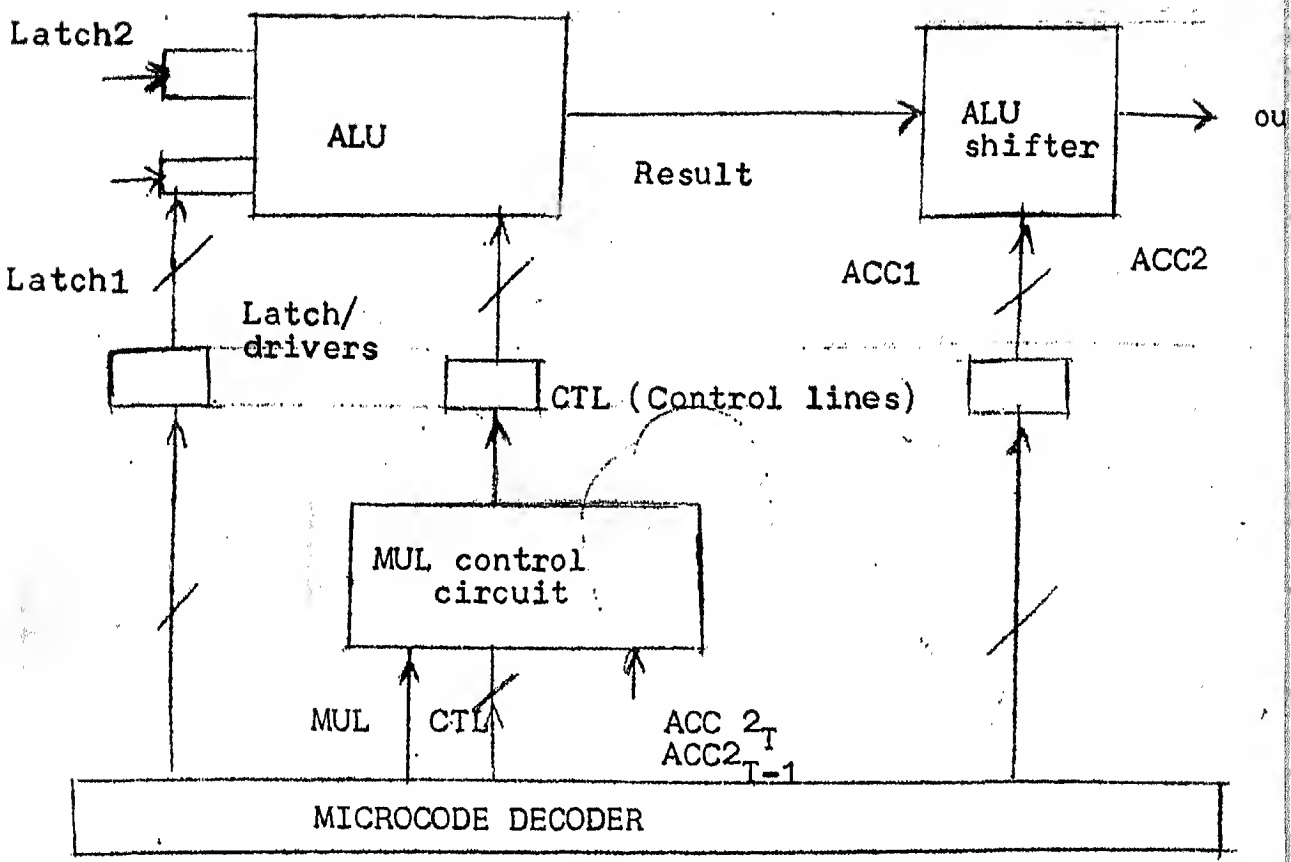


Fig. 4.15

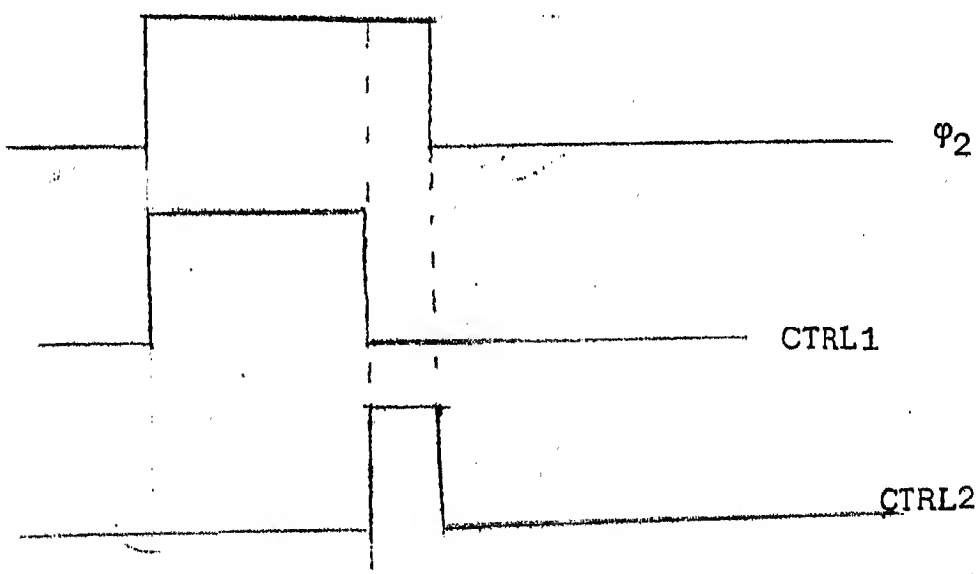


Fig. 4.12

TRUTH TABLE FOR MULTIPLICATION

MUL	ACC_{2T}	ACC_{2T-1}	PG_0	PG_1	PG_2	PG_3	KL_0	KL_1	KL_2	KL_3	R_0	R_1	R_2	R_3	C_{in}
0				NORMAL INSTRUCTIONS											
1	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0
1	1	1	0	1	1	0	0	0	0	0	0	1	1	0	0
1	0	1	0	1	0	1	1	0	0	0	0	1	0	1	0
1	1	0	1	0	1	0	0	0	0	1	0	1	0	1	1

Fig. 4.27

LAYOUT STRATEGY : SOME OBSERVATIONS

It is mentioned that diffusion can also be used as an interconnect, alongwith polysilicon and metal. But, it is useful to make use of the diffusion interconnect only when it is absolutely necessary. This is because of the large capacitance and resistance associated with the diffusion interconnect.

One more point is that the same column line may carry two different control signals, in two different parts of that line. This helps in minimizing the area.

In the NMOS technology, the length to width ratio of the pull up is often greater than that of the pull down. In our design, there are two types of pull up to pull down ratios. One is 4:1 and the other is 8:1. The channel length of the pull up is made to be greater than that of pull down. The widths of pull up and pull down are adjusted to achieve the desired length to width ratio. The gate of the pull up is always connected to its source. This is not explicitly shown in the stick figure.

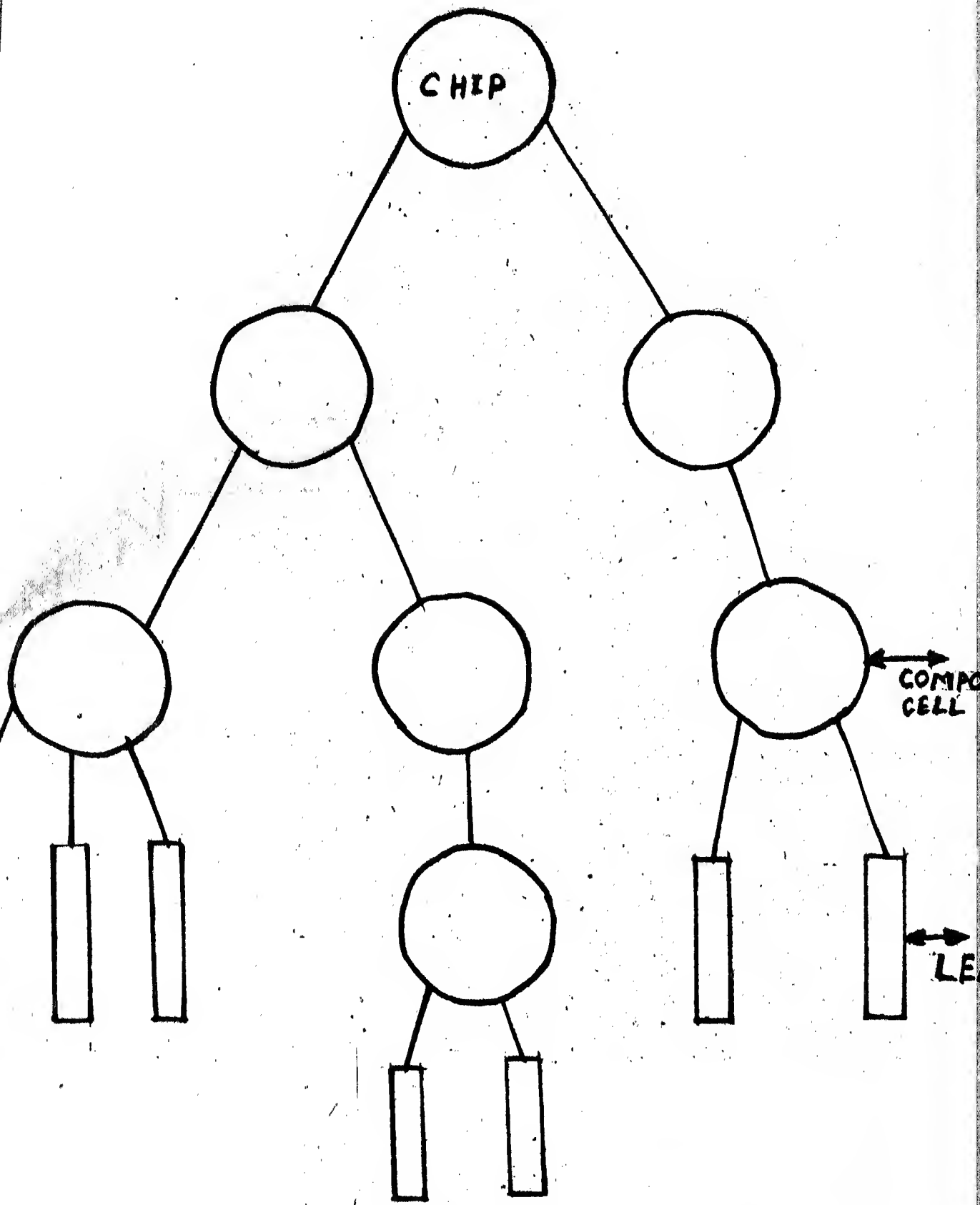


FIG. 1.1

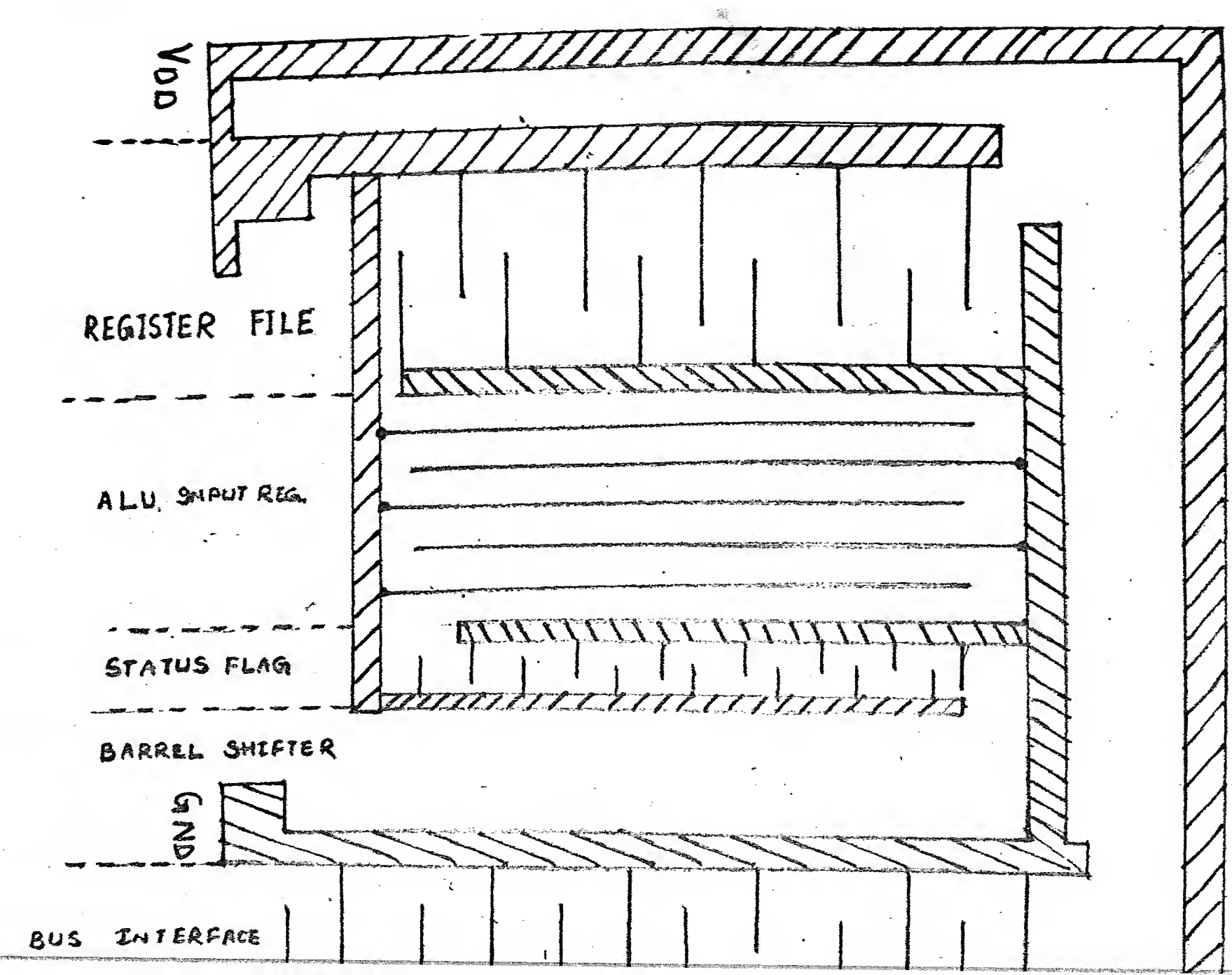
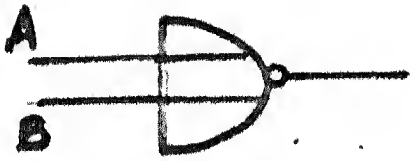
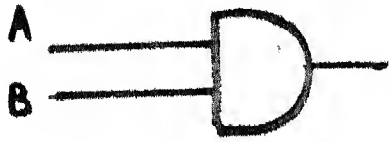

VDD
REGISTER FILE
ALU. INPUT REG.
STATUS FLAG
BARREL SHIFTER
GND
BUS INTERFACE

FIG 3.3



$\overline{A+B}$

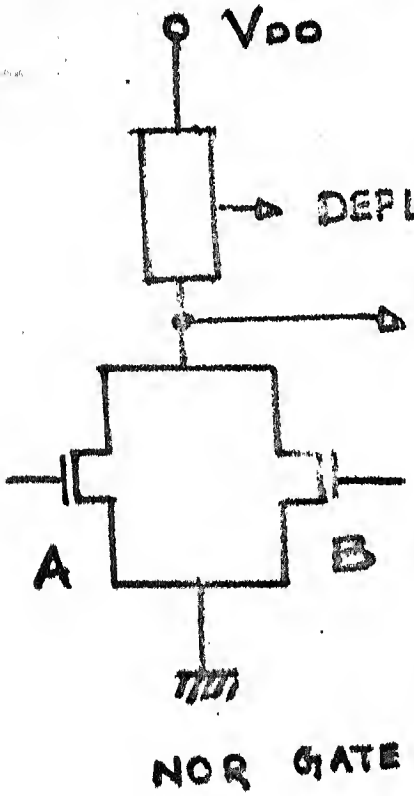
NOR GATE



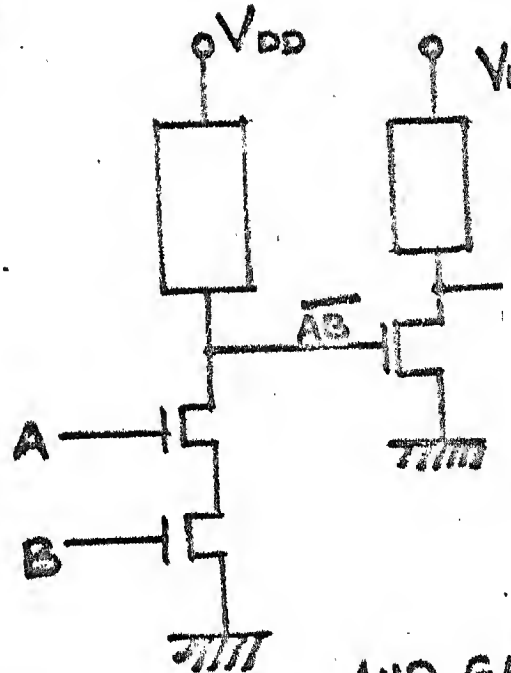
AB

AND GATE

NMOS REALISATION OF NOR AND AND GATE

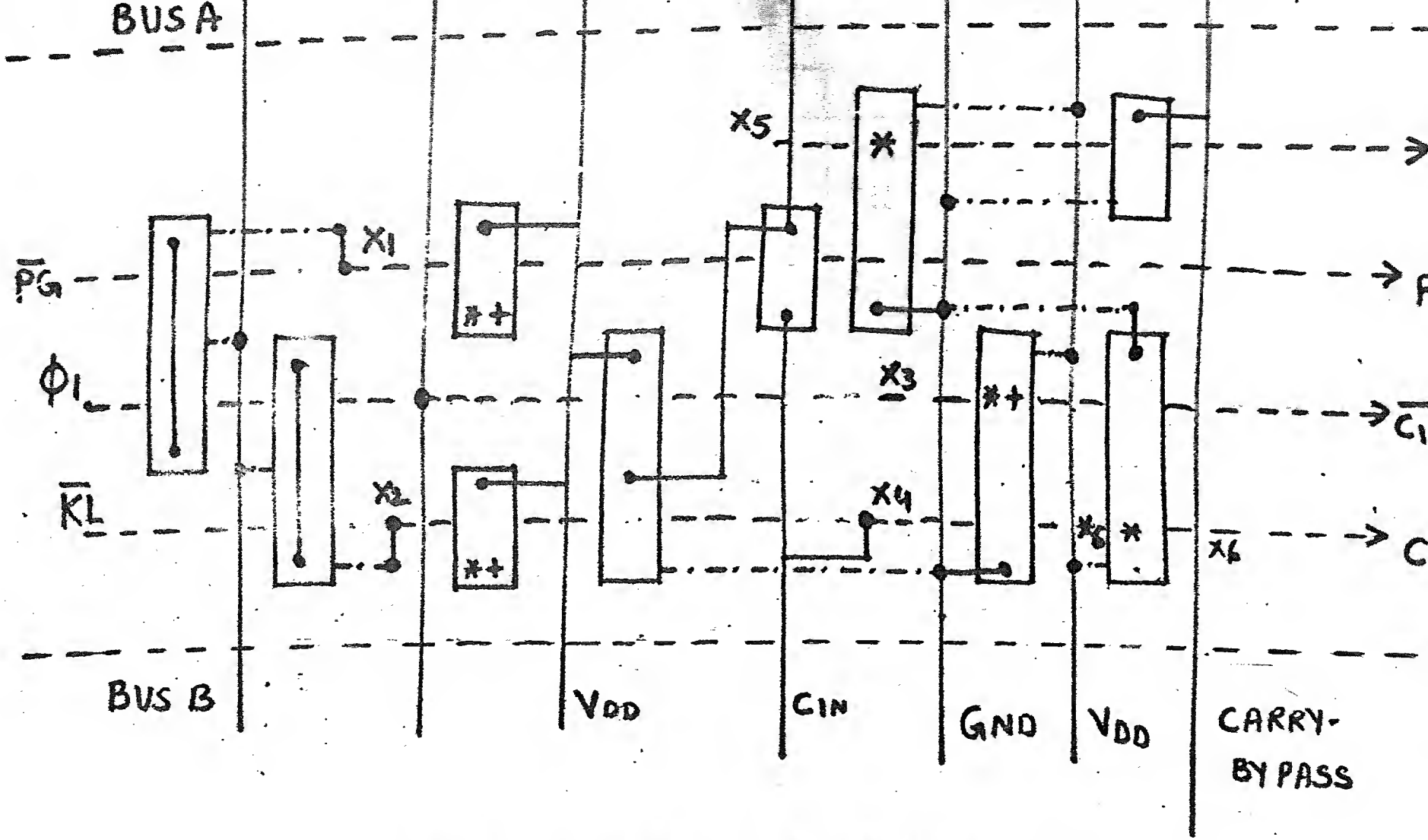


NOR GATE



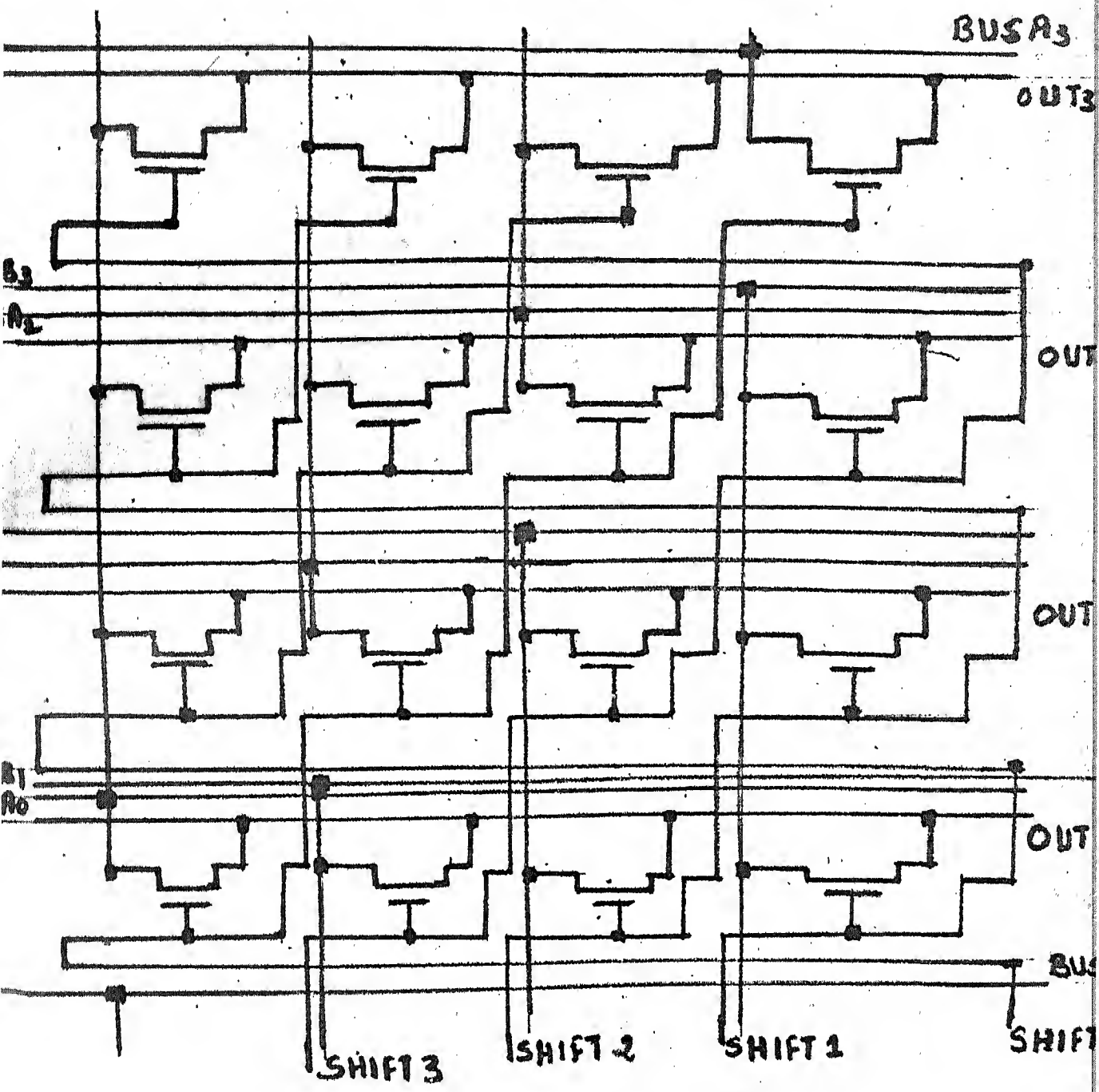
AND GAT

FIG : 4.19



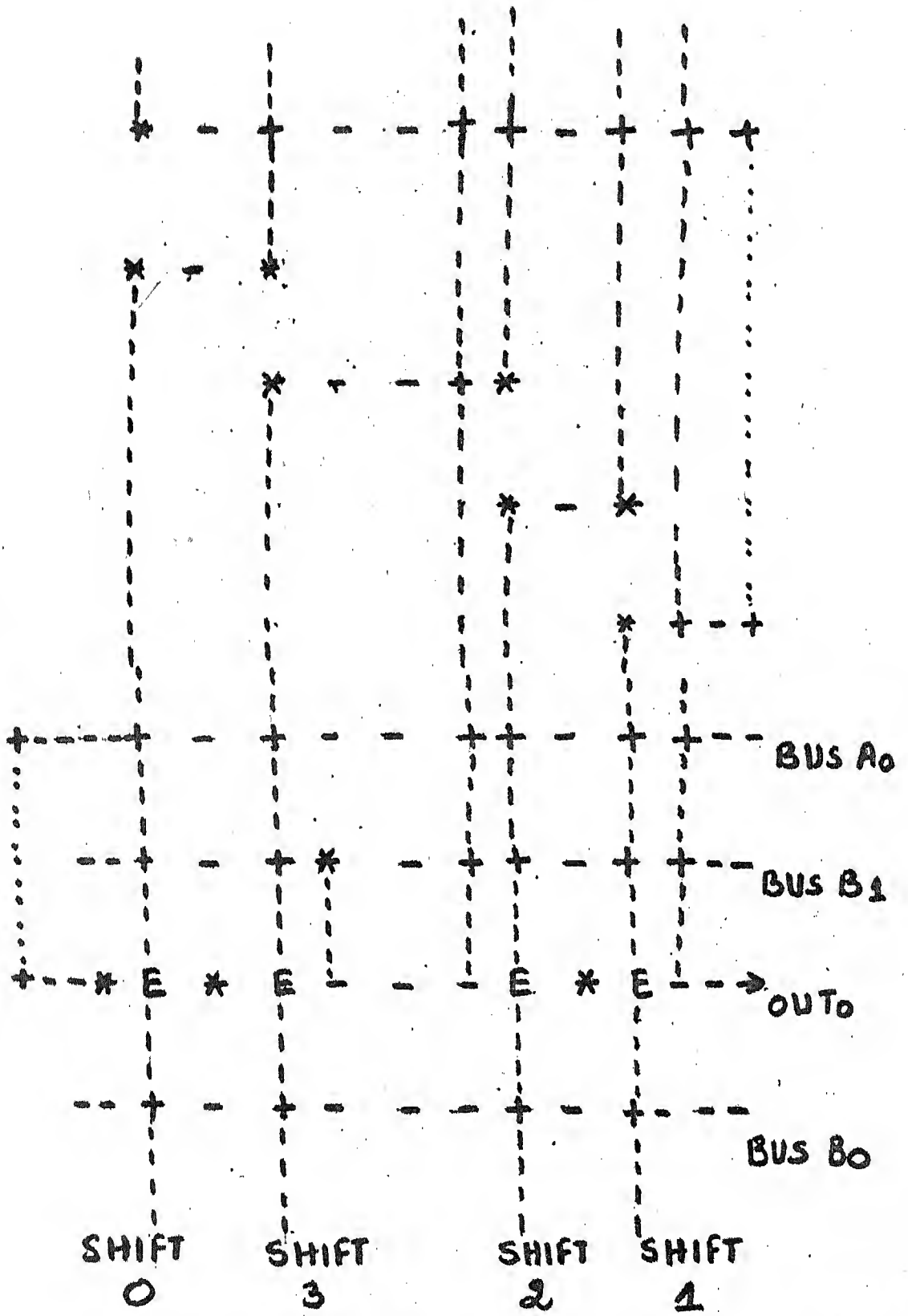
CARRY CHAIN - STICK FIGURE

FIG 3.19



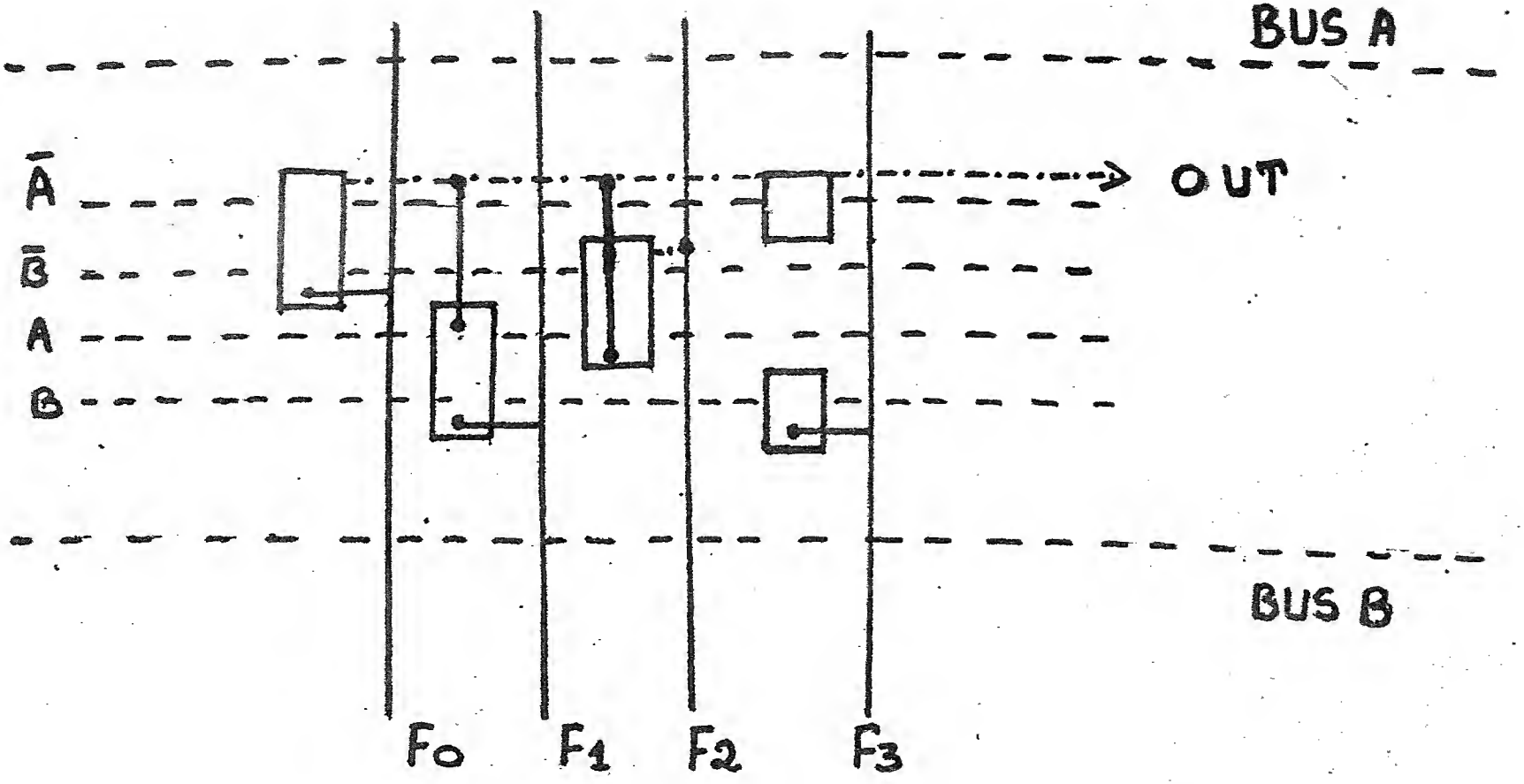
BARREL SHIFTER

FIG 4.5



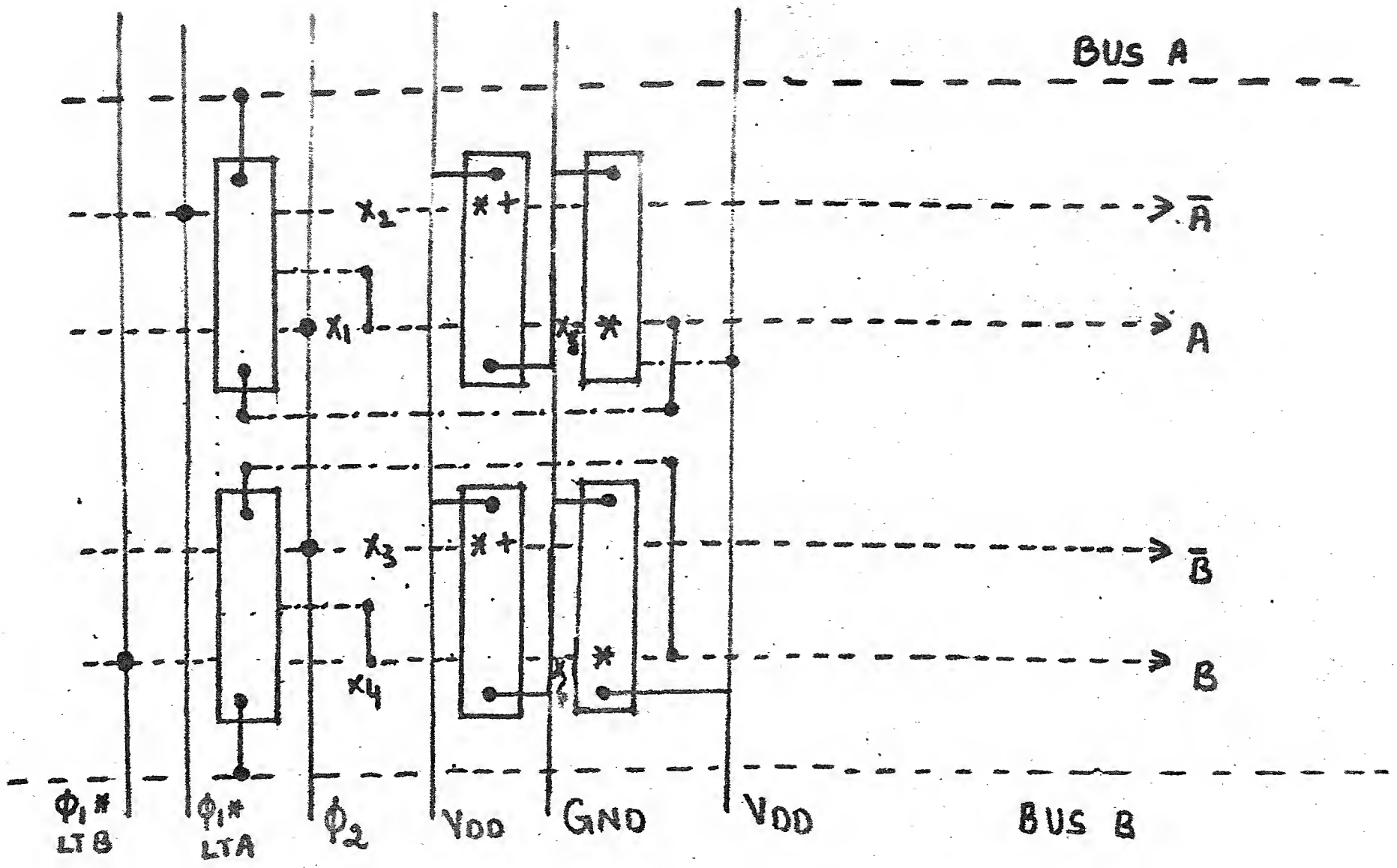
BARREL SHIFTER - DIGITIZED STICK FIGURE

FIG 4.11

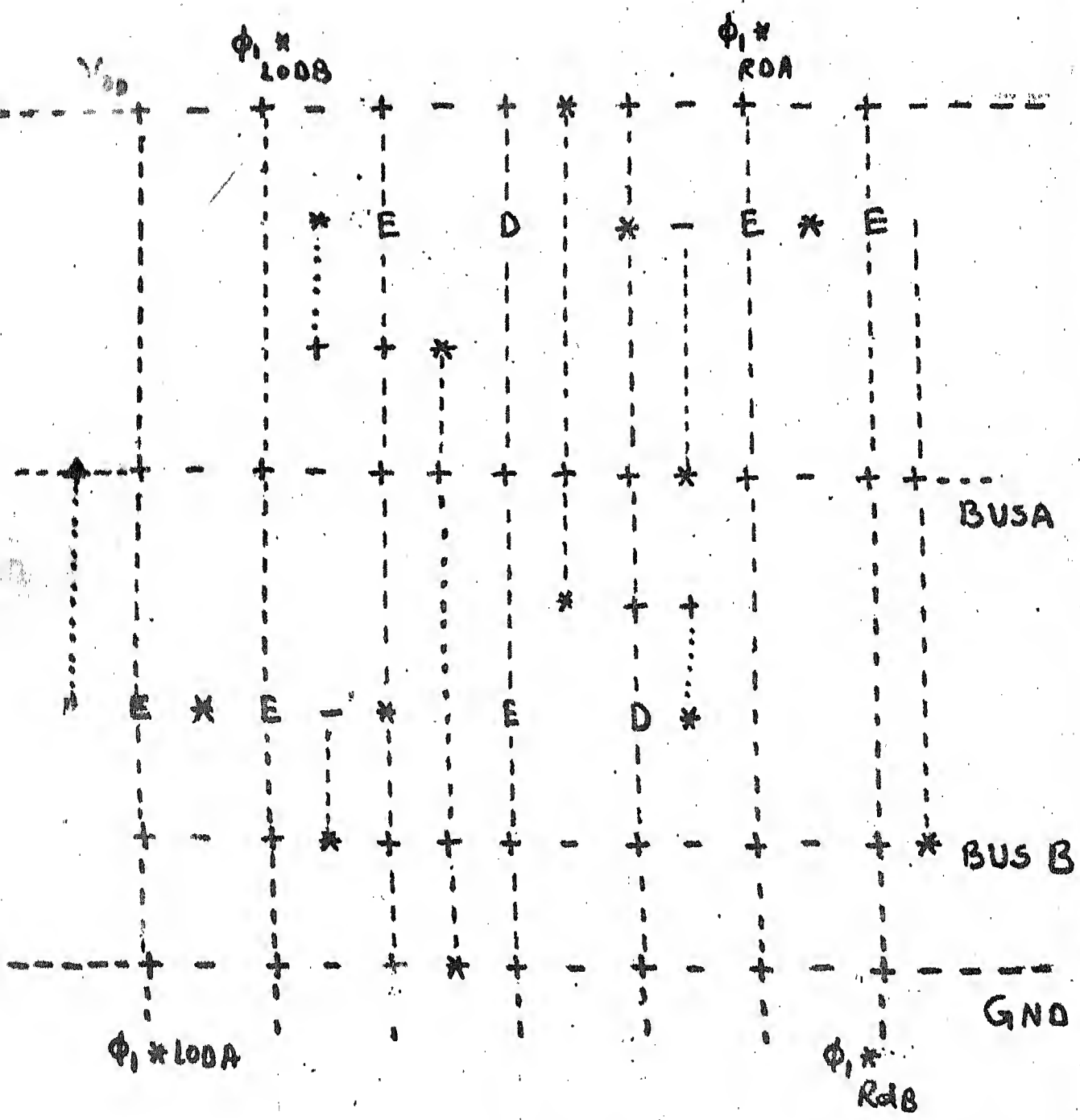


FUNCTION BLOCK STICK FIGURE

FIG 3.21

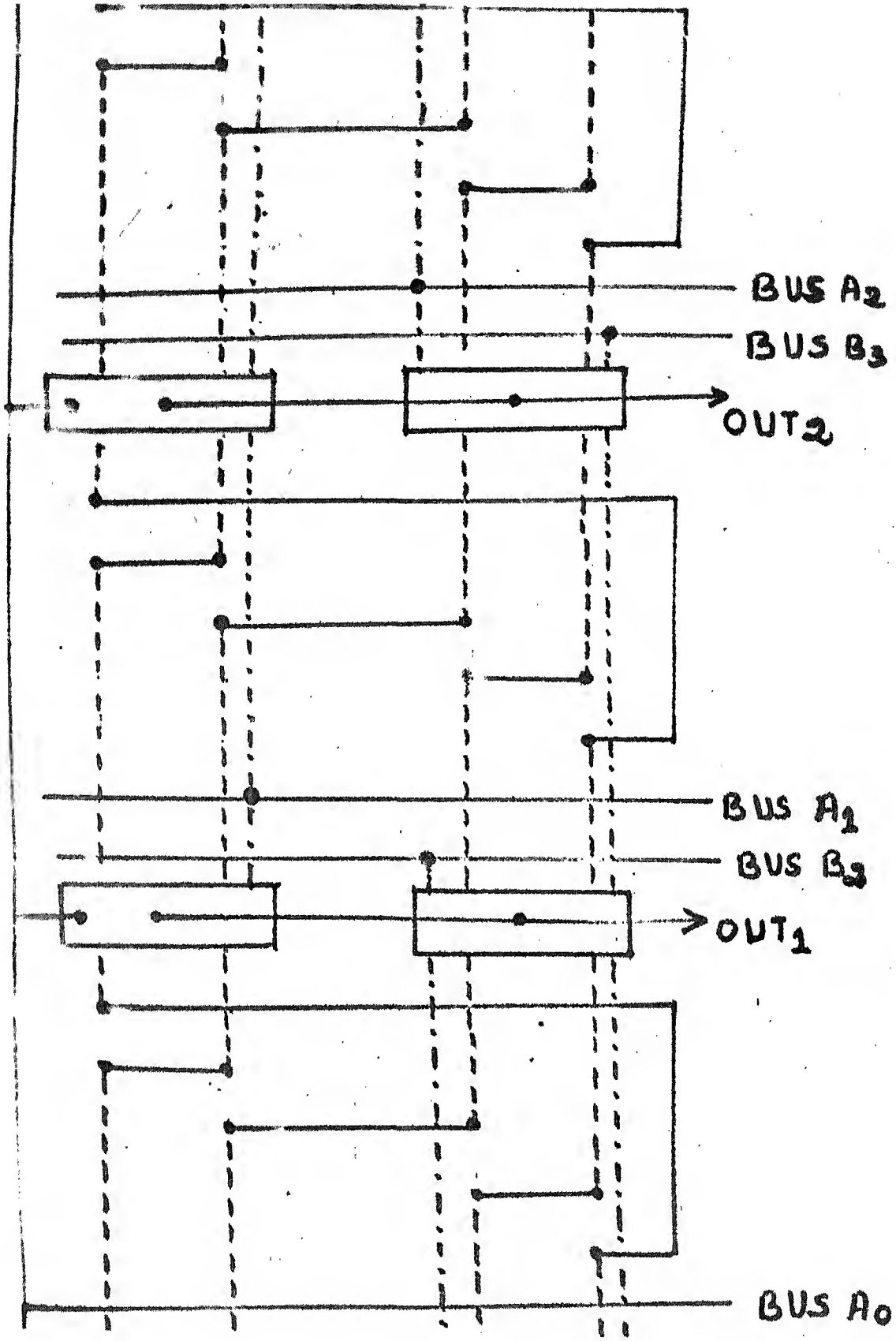


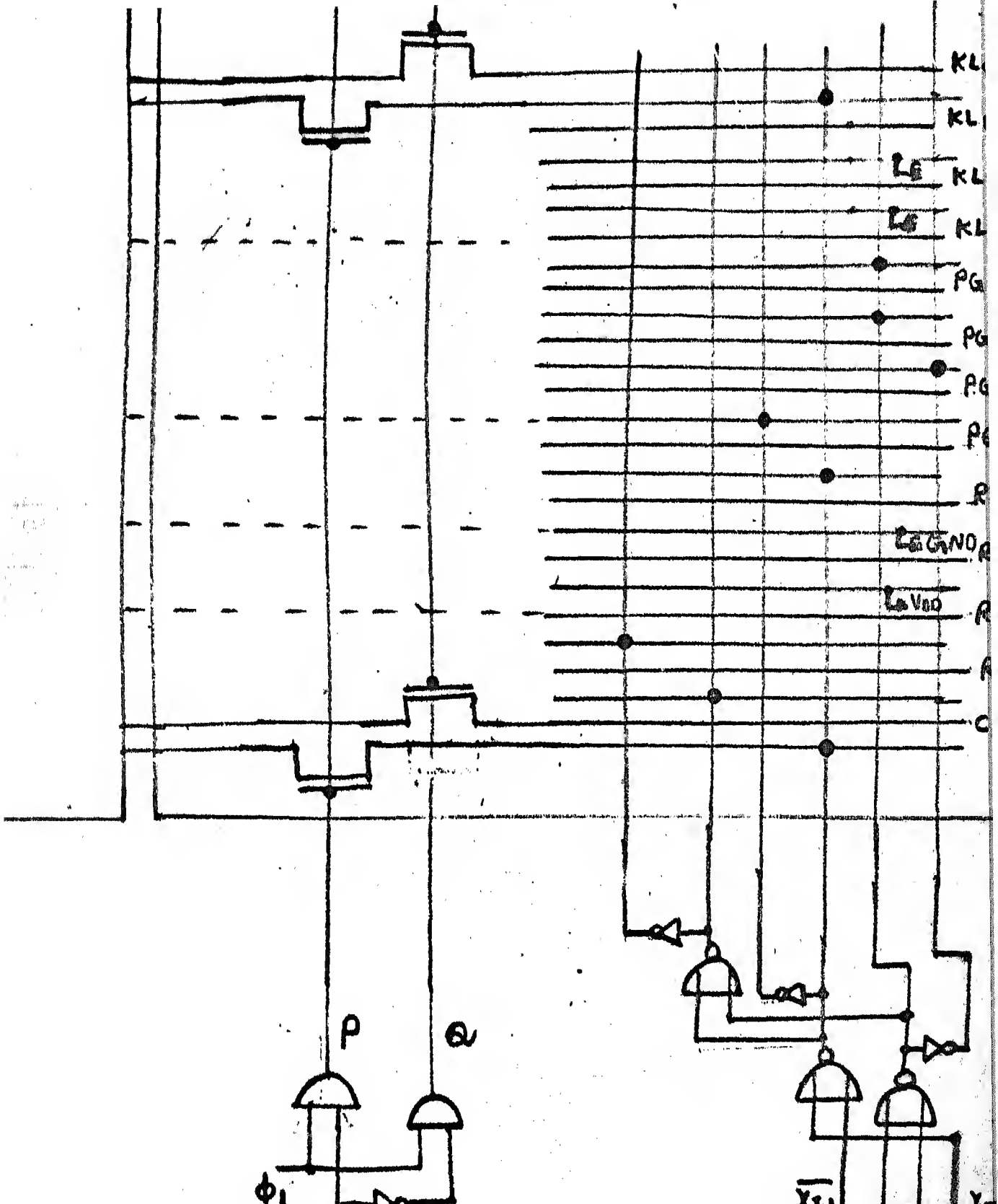
ALU INPUT REGISTER. STICK FIGURE



REGISTER CELL - DIGITIZED STICK FIGURE

FIG 4.4


BUS A_2
BUS B_3
OUT_2
BUS A_1
BUS B_2
OUT_1
BUS A_0



CONTROL CKT FOR MULTIPLICATION

EE - 1983- M- BAB- DES